DEEP EXPLORATION THROUGH PREIMAGE STATE DISTRIBUTION MATCHING (PIM)

Anonymous authors
Paper under double-blind review

ABSTRACT

We present an algorithm for reinforcement learning that explores and learns to solve long-horizon, goal-reaching tasks without any environment rewards, demonstrations, curriculum, or prior knowledge. With our approach, a simulated robot can learn to construct complex 8-cube configurations, a task not previously solved by prior methods based on intrinsic motivation or subgoal selection. We show that the generalization capabilities of generative models *trained from scratch* can drive exploration, even in difficult, high-dimensional tasks. We train a generative model with only the online data collected by the policy to predict what states are likely to precede a goal state. Before the goal state been reached, this process is analogous to prompting an image generation model on an unseen prompt. Initially, this generative model makes poor predictions, but when the policy tries and fails to visit those states, it collects the data required to fix the model's predictions errors. Our algorithm outperforms a diverse set of baselines across multiple long-horizon tasks involving robotic manipulation, construction, and navigation. On the AllegroHand dexterous grasp-and-throw task, even the best-performing baseline does not obtain reliable success despite using $2\times$ more interactions.

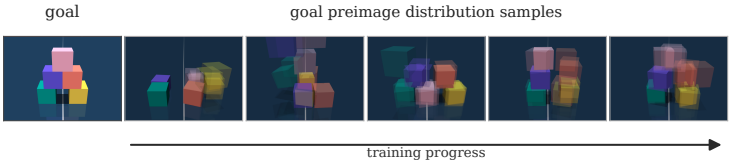


Figure 1: **Preimage Distribution Matching.** A generative model predicts the states that precede the goal, and a policy is optimized to match this distribution. Importantly, the model's predictions change throughout training, driving exploration.

1 INTRODUCTION

The conventional wisdom in reinforcement learning (RL) today is that learning from scratch is impossible; thus, exploring an environment from scratch is misguided, and we must rely upon foundation models or prior datasets to guide exploration (Ye et al., 2025; Nair et al., 2023; Ma et al., 2023). Simultaneously,generative models are capable of generating images and text when given prompts unseen during training (e.g., "an astronaut riding a horse on the moon" (Ramesh et al., 2022; Saharia et al., 2022)). This raises the question: *can we utilize the generalization capabilities of generative models to predict which states would be visited if the task were to be solved, and use those predictions to drive exploration in RL?* This exploration paradigm is advantageous because it remains tractable in high-dimensional state spaces — it does not require exploring the entire environment and instead exploits *task structure* to narrow down the space of plausible states. As long as the generative model can infer global information about the task from local observations, it can guide goal-directed data collection. This approach stands in contrast to methods based on upper-confidence-bound bonuses (Azar et al., 2017; Jin et al., 2018) or bonuses that encourage novelty, high state or action entropy, or information gathering (Burda et al., 2019; Pathak et al., 2017; Houthooft et al., 2016; Haarnoja et al., 2018; Agarwal et al., 2023), which are designed to broadly explore unseen states and they can solve *arbitrary unstructured* tasks as long as the state space size is not large (regret bound polynomial in state space size). Our experiments reflect the same: state-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

of-the-art novelty-seeking methods can solve three-cube block-building tasks, but they fail as the number of blocks increases to 7 and 8.

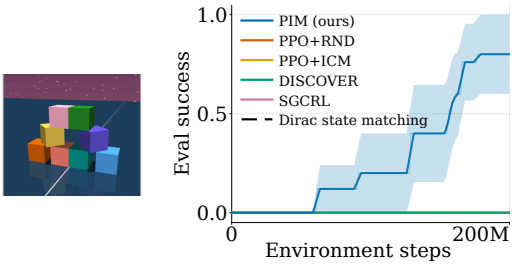


Figure 2: Preimage matching (PIM) solves 8 cube block building tasks that no intrinsic motivation method or prior goal-conditioned RL method can solve.

Similar to the way that generative models generate images for unseen prompts, in this paper *we use generative models to predict the distribution over states that precede an unseen goal state*, i.e., the *goal-preimage distribution*. We call our approach **PreImage Matching** (PIM). By optimizing a policy to match this preimage distribution, we obtain a policy that automatically exhibits deep exploration, yet asymptotically converges to a deterministic goal-reaching policy. Data collected by the policy is then used to further train the same generative model that models the occupancy measure, resulting in a better estimate of the preimage distribution. Exploration emerges as a consequence of using a learned reward; our method employs neither exploration bonuses and nor uncertainty quantification. While state distribution matching objectives are typically challenging to optimize in general, we show that our objective is equivalent, up to first-order policy gradients, to a reward-maximization problem with reward $r(s) = \log p^{\pi}_{\gamma}(g \mid s)$, hence a simpler distribution matching objective than prior works (Agarwal et al., 2023; Durugkar et al., 2021). Empirically, we show that maximizing the goal log-likelihood reward using Proximal Policy Optimization (PPO) (Schulman et al., 2017) achieves state-of-the-art performance on hard exploration tasks. Our experiments demonstrate that this general recipe is applicable to many choices of density estimators, both TD and MC, with some of the strongest results coming from normalizing flows.

Our contributions are as follows:

- We introduce goal-preimage state distribution matching (PIM), which uses a state density estimator to estimate the distribution of states preceding the goal and matches this distribution using an f-divergence. We show that this process yields monotonic policy improvement and converges to the optimal goal-reaching policy under standard assumptions.
- We obtain the policy gradient of our objective and show its equivalence to a dynamic reward maximization problem with log-likelihood rewards, providing a practical algorithm for goal-reaching.
- Our method achieves state-of-the-art results on challenging long-horizon tasks including BuilderBench (Ghugare et al., 2026), a Meta-World hard exploration peg-insertion task (Yu et al., 2020), the grasp-and-throw task from the Allegro-Kuka (23 DoF) environment in Isaac Gym (Makoviychuk et al., 2021) and a ManiSkill combined navigation and manipulation task (Gu et al., 2023). Our method solves 7-cube and 8-cube block building tasks never previously solved by state-of-the-art baselines based on intrinsic rewards, including RND and ICM (Burda et al., 2019; Pathak et al., 2017), subgoal selection (Diaz-Bone et al., 2025), contrastive goal-conditioned RL (Liu et al., 2025) and prior state matching methods (Agarwal et al., 2023). On the AllegroHand task, our method starts achieving reliable success after only 200M environment steps, while even after 400M interactions, the best baseline does not reach reliable success. To the best of our knowledge, *PIM is the first method to highlight how the generalization capabilities of generative models alone suffice to drive exploration in long-horizon tasks without the need for ensembles, uncertainty modeling, or intrinsic reward.*
- Our approach is compatible with any density estimator. Experiments using both TD and MC methods for estimating occupancy measure show that a wide range of occupancy estimation methods can support exploration guided by generalization.

2 Related Works

Probabilistic modeling and Reinforcement learning. Reinforcement learning (RL) and generative modeling are inextricably linked: generative models play an important role in modern RL algorithms, e.g., as policies (Wang et al., 2023), value distributions (Bellemare et al., 2017), and world models (Hafner et al., 2020). In fact, RL agents themselves can be understood as generative

models and trained via divergence minimization (Codevilla et al., 2018; Mandlekar et al., 2018; Ho & Ermon, 2016; Fu et al., 2018; Ghasemipour et al., 2020; Finn et al., 2016). For instance, behavioral cloning (BC), the most popular imitation learning method, matches the expert action distribution (Codevilla et al., 2018; Mandlekar et al., 2018), and inverse RL has been shown to be equivalent to minimizing the divergence between the policy and expert occupancy measures (Ho & Ermon, 2016; Fu et al., 2018; Ghasemipour et al., 2020; Finn et al., 2016). This perspective has motivated methods that treat RL as a distribution matching task. Closest to our work, Lee et al. (2019); Durugkar et al. (2021) match the policy occupancy measure to the goal distribution (delta Dirac distribution at goal). However, these methods are limited because they fail to solve long-horizon goal-reaching tasks (see Sec. 5.5).

Exploration with generative models. Much of the interest in the intersection of RL and generative models comes from a hope of addressing exploration in a principled way (Ziebart et al., 2008; Levine, 2018; Kappen et al., 2012): generative models could measure uncertainty to create intrinsic rewards (Bellemare et al., 2016; Houthooft et al., 2016), and policies could be optimized to match broad (e.g., uniform) state (Lee et al., 2019) or action distributions (Haarnoja et al., 2018) to drive exploration. However, probabilistic approaches like Maximum-entropy RL (Haarnoja et al., 2018) target only aleatoric uncertainty; they encourage action stochasticity, which is insufficient for challenging long-horizon exploration (Liu et al., 2025). In contrast, Bayes-adaptive methods that capture epistemic uncertainty are intractable (Osband et al., 2013; Kolter & Ng, 2009) for practical tasks. Our analysis (App. B.5) shows a connection between an instantiation of our method and an approximate (tractable!) Bayesian approach that estimates the distribution over states preceding the goal, rather than a distribution over transition functions. Prior practical versions of epistemic uncertainty methods such as RND (Burda et al., 2019) and other novelty seeking methods (Burda et al., 2019; Pathak et al., 2017; Zhang et al., 2021; Houthooft et al., 2016; Seo et al., 2021; Mohamed et al., 2026) fail in high-dimensional state and action spaces (Ghugare et al., 2026; Ecoffet et al., 2021; Mhammedi & Cohan, 2026), since they do not perform task-directed exploration and instead encourage visiting all novel states. However, our method performs goal-directed exploration even in early training; therefore, it can solve problems in high-dimensional state spaces. Refer to App. A for more related works.

3 PRELIMINARIES

Notation. We consider a Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, g, \gamma, \rho)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, $P(s' \mid s, a)$ is the transition probability, and $g \in \mathcal{S}$ is the goal state. We use discount factor $\gamma \in (0, 1)$ and initial state distribution $\rho(s)$.

Discounted occupancy measure. Discounted occupancy measure is the on-policy distribution of a future state S_T, where T is sampled from a geometric distribution $\text{Geom}(1-\gamma)$ (Eysenbach et al., 2022). Conditioned on an initial state-action (s, a), the discounted future-state occupancy measure is defined as (For our analysis, it is useful to define occupancy measures in which future timesteps are weighted according to other distributions; see App. B.1.)

$$p^\pi_\gamma(s_f \mid s, a) = (1-\gamma) \sum_{t=0}^{\infty} \gamma^t \overset{\pi}{\Pr}(S_t = s_f \mid S_0 = s, A_0 = a). \tag{1}$$

Occupancy measures are useful because a general discounted reward-maximization objective can be expressed directly in terms of the discounted occupancy measure:

$$Q^\pi(s, a) \coloneqq \mathbb{E}^\pi \left[\sum_{t=0}^{\infty} \gamma^t r(S_t) \,\middle|\, S_0 = s, A_0 = a \right] = \frac{1}{1-\gamma} \int_{\mathcal{S}} p^\pi_\gamma(s_f \mid s, a) r(s_f)\, ds_f.$$

In particular, consider the sparse-reward goal-reaching setting with reward $r(s) = \mathbb{1}\{s = g\}$. Then, $Q^\pi(s, a) = \frac{1}{1-\gamma} p^\pi_\gamma(g \mid s, a)$ (Eysenbach et al., 2022). Therefore, maximizing the expected discounted sparse reward is equivalent to maximizing the discounted occupancy of the goal state. With a slight abuse of notation, we use $p^\pi_\gamma(s)$ to denote the marginal discounted state occupancy measure induced by policy π under the initial state distribution ρ. That is,

$$p^\pi_\gamma(s) \coloneqq \mathbb{E}_{s_0 \sim \rho,\, a_0 \sim \pi(\cdot \mid s_0)} \left[p^\pi_\gamma(s \mid s_0, a_0) \right]. \tag{2}$$

Our objective is to find a policy that maximizes the occupancy of the goal:

$$\max_{\pi} p_\gamma^\pi(g). \tag{3}$$

Density estimation with Normalizing Flows. The method we introduce relies on an underlying generative model. Our experiments will primarily use normalizing flows, though we will demonstrate that our method can readily be applied to other density models. Appendix B.6 provides a brief primer on the objectives for normalizing flows.

4 Method: Distribution Matching to the Goal's Preimage

The key idea behind our method is to cast goal reaching as a distribution matching problem on occupancy measures. But, rather than trying to align a policy's occupancy measure with a goal state, we will align a policy's occupancy measure with the (geometrically-distributed) *preimage* of the goal. As we will show in Sec. 4.1 this objective corresponds to an evidence lower bound on the goal-reaching problem, with intermediate states as a latent variable. We show how this occupancy matching objective is equivalent to a reward maximization problem with dynamic reward. In Sec. 4.2 we outline the building blocks of our algorithm and its implementation recipe.

4.1 Variational lower bound

The key ingredient in our method is the *preimage* of the goal state. Whereas the discounted state occupancy measure (Eq. 1) indicates what happens in the (time-averaged) future, the pre-image is an analogous quantity that indicates what happens in the (time-averaged) past. After formally defining this preimage distribution, we will show how the goal-reaching objective (Eq. 3) can be related to the pre-image distribution. Importantly, by estimating this distribution with function approximation, we can make predictions about what states *would* precede the goal before the goal has ever been visited. This will provide a dense training signal to the policy, providing an automatic form of exploration.

Definition 1. *Let $p_\gamma^\pi(s_f \mid s)$ be the conditional and $p_\gamma^\pi(s)$ the marginal occupancy measure then:*

$$p_\gamma^\pi(s \mid s_f = g) := \frac{p_\gamma^\pi(s_f = g \mid s)\, p_\gamma^\pi(s)}{Z} \tag{4}$$

is the goal-preimage state distribution, with normalization constant $Z := \int p_\gamma^\pi(s_f = g \mid \hat{s})\, p_\gamma^\pi(\hat{s})\, d\hat{s}$.

Preimage distribution weights states according to how likely they are to have been visited before reaching the goal, with larger weights assigned to states visited closer to the goal. We can use a variational lower bound to relate preimage distribution to goal reaching objective:

$$\begin{aligned} \log p_{\bar{\gamma}}^\pi(s_f = g) &= \log \int p_\gamma^\pi(s_f = g \mid s) p_\gamma^\pi(s)\, ds \\ &\geq \underbrace{\mathop{\mathbb{E}}_{q(s)}\left[\log p_\gamma^\pi(s_f = g \mid s)\right]}_{\text{backward consistency}} - \underbrace{D_{KL}(q(s) \| p_\gamma^\pi(s))}_{\text{forward consistency}} = \mathcal{J}(\pi, q) \end{aligned} \tag{5}$$

Refer to Appendix B.2 for the derivation of the ELBO. The notation $\bar{\gamma}$ is used to denote a slightly different discounting than the standard geometric discounting (details in App. B.1). Intuitively, this objective suggests that, in order to maximize the goal-reaching probability $p_\gamma^\pi(s_f = g)$, we should alternate between optimizing a variational state distribution $q(s)$ and the policy occupancy measure $p_\gamma^\pi(s)$. The distribution $q(s)$ should place more weight on states that are likely to be visited before reaching the goal (backward consistency). At the same time, it should remain close to the forward occupancy measure induced by the old policy (forward consistency). Applying the methods of Lagrangian shows that the optimal $q(s)$ is the goal preimage distribution:

$$q^*(s) = p_\gamma^\pi(s \mid s_f = g) \tag{6}$$

According to the second ELBO term, the policy is then updated so that its occupancy measure is as close as possible to $q^*(s)$ under the reverse KL divergence (See App. B.2 to see why we can ignore the dependency of the first ELBO term on policy π during policy optimization). Thus we define our policy optimization problem as:

$$\min_{\pi} D_f\left(p_\gamma^\pi(s) \,\middle\|\, p_\gamma^{\pi_{\text{old}}}(s \mid s_f = g)\right). \tag{7}$$

where D_f denotes an f-divergence (reverse KL divergence obtained from the ELBO is a special case), and π_{old} denotes the current policy to which the update is applied. Objective 7 isn't a standard reward-maximizing policy objective, and therefore the policy-gradient theorem (Sutton et al., 1999a) does not apply. However, in App. B.3, we show that the gradient of Objective 7 with respect to the policy parameters θ, evaluated at θ_{old}, is exactly equal to the gradient of a standard reward-maximization objective. For example, when D_f is the forward KL divergence, the corresponding reward is simply the goal log-likelihood, $r(s) = \log p_\gamma^{\pi_{\text{old}}}(s_f = g \mid s)$. For the choice of forward KL divergence, this reward-maximization equivalence can be seen more directly by approximating $p_\gamma^\pi(s) \approx p_\gamma^{\pi_{\text{old}}}(s)$:

$$\max_\pi -D_{\text{KL}}(p_\gamma^\pi(s) \| p_\gamma^\pi(s_t \mid s_f = g)) = \mathbb{E}_\pi[\sum_t \gamma^t(-\log p_\gamma^\pi(s_t) + \log p_\gamma^\pi(s \mid s_f = g))]$$

This is a reward maximization with $r_t = -\log p_\gamma^\pi(s_t) + \log p_\gamma^\pi(s_t \mid s_f = g)$. We plug in the preimage definition: $r_t \overset{4}{=} \log p_\gamma^\pi(s_f = g \mid s_t) + \cancel{\log p_\gamma^\pi(s_t)} - \log Z - \cancel{\log p_\gamma^\pi(s_t)}$. So, the reward function can be obtained by simply estimating the density $\log p_\gamma^\pi(s_f = g \mid s_t)$ (the Z term is a constant w.r.t. the state, so dropping it from the reward does not change the optimal policy). Thus, we can directly use the log likelihood as a reward function and obtain the optimal policy by solving this reward maximization problem. So overall, Bayes' rule helps to turn the *backward* (preimage) distribution matching into maximizing a *forward* reward. In App. B.4 we show that our method is monotonic policy improvement under standard assumptions and that its time complexity improves upon advantage-based policy improvement (Kakade & Langford, 2002).

Finally, note that although on the surface our variational ELBO derivation resembles prior inferential-based approaches to control (Haarnoja et al., 2018; Abdolmaleki et al., 2018; Levine, 2018), there are several important differences. MaxEnt RL methods (Haarnoja et al., 2018; Ziebart et al., 2008; Nachum et al., 2017) treat the trajectory as the unobserved random variable with a uniform prior, and are not particularly effective at exploration (Farrahi & Mahmood, 2026; Yang et al., 2025). The root cause is the uniform prior assumption, which encourages high-entropy behavior even asymptotically. Prior approaches (Abdolmaleki et al., 2018; Levine, 2018) cast RL as an inference problem over *actions*; our method instead casts the problem as inference over *states*. In App. C.1.1, we empirically show that state matching is significantly more effective and in App. B.5 we show that under certain parametrization and assumptions preimage distribution resembles an approximate Bayesian method that captures epistemic uncertainty about states preceding the goal.

4.2 Our complete algorithm: Preimage State Distribution Matching (PIM)

The core of our algorithm is a conditional state-density estimator that estimates $\log p_{\gamma,\phi}(s_f \mid s, a)$. This model is trained on (s, a) sampled from the replay buffer, with s_f sampled from the corresponding geometric time-discounted future-state distribution. As training progresses and the policy becomes better at collecting more goal-relevant data, the density estimator receives better training data. For policy optimization, we only use the log likelihood estimation evaluated at the task goal, i.e., $\log p_{\gamma,\phi}(g \mid s, a)$ as the reward. However, training the density estimator using future states s_f sampled from the replay buffer is essential, since it allows the model to learn the structure of the observed data and generalize to the unseen goal g even before the goal has been visited (see Fig. 6 for directed exploration even before seeing the goal). The density estimator can be any probabilistic modeling method that provides a likelihood or density-ratio estimate, such as normalizing flows (NF) or contrastive learning (CL) or TD methods. In our experiments, we found NF most useful; the training objective of NF is as follows (See App. B.6):

$$\max_\phi \mathbb{E}_{s_f \sim p_\gamma^\pi(.|s,a)}[\log p_{\gamma,\phi}(s_f \mid s, a)] \tag{8}$$

Preimage Matching (PIM)

Our algorithm alternates between doing RL with the density model as a reward and updating the density model with maximum likelihood using the collected data:

$$r(s_t, a_t) = \log p_{\gamma,\phi}(s_f = g \mid s = s_t, a = a_t)$$

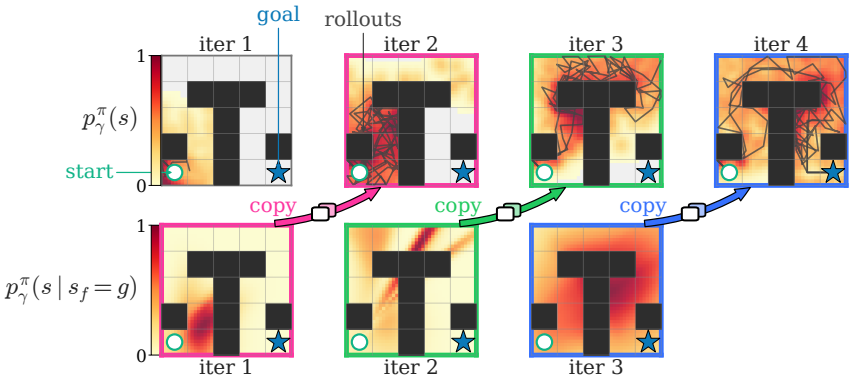


Figure 3: Preimage shows the agent's belief about states preceding the goal. Even before seeing the goal, the preimage is helpful for guiding exploration. The agent's incorrect belief is corrected by collecting more data.

For policy optimization, we use an on-policy policy-gradient method, PPO (Schulman et al., 2017). Using a policy-gradient method is important because our theoretical analysis in App. B.3 shows that preimage distribution matching is equivalent to the surrogate reward-maximization objective only up to first order. We found that smoothing the log-likelihood with respect to the state, i.e., regularizing $\mathbb{E}\big[\,\|\nabla_s \log p_{\gamma,\phi}(s_f \mid s, a)\|_2\,\big]$, significantly helps with the monotonicity of the reward. We use dual optimization to apply this regularization (See App. B.6 and Sec. 5.4). Unlike prior works (Sermanet et al., 2018; Ma et al., 2023; 2025; Adamczyk et al., 2025; Ye et al., 2025; Ma et al., 2023) that use reward functions or distances learned from expert data, our approach learns the reward from scratch (no prior data) and changes dynamically as the policy explores.

5 Experiments

- **Q1** Does the density model make useful predictions when evaluated at the unseen goal? (Sec. 5.1)
- **Q2** Does our algorithm (PIM) work in challenging long-horizon goal-reaching settings? How is it qualitatively different from other exploration methods? (Sec. 5.2)
- **Q3** Is the choice of density estimator important for the performance of PIM? Do we see emergent exploration across different density estimator objectives? (Sec. 5.3)
- **Q4** Is any regularization required to make the estimated log-likelihoods a smooth and monotonic reward, suitable for policy training? What other training choices for the density estimator are important in obtaining useful rewards? (Sec. 5.4)
- **Q5** How is PIM different from state distribution matching with goal distribution $\delta_g(s)$? (Sec. 5.5)

5.1 Preimage evolution before seeing the goal (Q1)

Figure 3 shows how the policy occupancy and preimage evolve in a maze. Initially, the preimage incorrectly places mass behind the wall (iter 1), and the policy follows it. The resulting data corrects this belief, shifting the preimage toward the upper hallway (iter 2). Subsequent policy updates follow the improved preimage until it concentrates on a path from the initial state to the goal (iter 3) and the policy reaches the goal (iter 4). Thus, the preimage acts as an evolving belief over states preceding the goal and incorrect beliefs generate data that progressively corrects them.



Figure 4: **Environments.** We evaluate PIM on long-horizon tasks without any rewards.

5.2 Baseline Experiments (Q2)

Benchmarks. We use four challenging long-horizon benchmarks. First, the peg-insertion task from MetaWorld (Yu et al., 2020); second, 8 block building tasks from (Yan, 2026) which is based upon builderbench benchmark (Ghugare et al., 2026). BuilderBench is specifically designed to test

open-ended exploration through building structures with blocks. Third, we use ManiSkill hybrid navigation-and-manipulation tasks (Gu et al., 2023). Finally, grasp-and-throw tasks in the Allegro-Hand environment from Isaac Gym (Makoviychuk et al., 2021). AllegroHand is a dexterous robotic hand with 23 degrees of freedom; because of this high dimensionality, prior work has studied it only with carefully designed dense reward functions (Handa et al., 2023; Petrenko et al., 2023). In all tasks, we don't use any dense or shaped reward other than the Normalizing flows (NF) log-likelihood estimates (Refer to App C.2 for more details about the tasks).

Baselines. We compare against five groups of baselines. (i) Intrinsic-motivation methods based on novelty seeking: we use PPO+RND (Burda et al., 2019) and PPO+ICM (Pathak et al., 2017)(ii) Subgoal-based GCRL methods: we use DISCOVER (Diaz-Bone et al., 2025), which selects subgoals based on novelty, reachability, and goal-similarity. (iii) Goal state matching: f-policy gradient that matches the policy occupancy to a fixed Dirac distribution concentrated at the goal (Agarwal et al., 2023). (iv) Single-goal contrastive RL (Liu et al., 2025): an off-policy GCRL method that explores well without any sub-goal selection mechanism, but requires training policies for all goals.

Results. For simpler tasks, the performance gap between PIM and the baselines is not significant. However, as the task horizon increases—for example, with more cubes in BuilderBench—the other methods stop achieving any success, while our method continues to perform well (Figure 7, 10). The main failure mode of RND and Dirac state matching is due to their pure novelty seeking and undirected exploration: they visit many states unrelated to the goal until they observe a sign of success. This can be effective in small environments, but stops working in high-dimensional environments. Figure 5 shows that, before the first success, RND has a constant distance to the goal and visits goal-irrelevant states. In contrast, PIM consistently approaches the goal, and states visited before reaching the goal already make partial progress. DISCOVER subgoal selection strategy is not effective in hard tasks. It selects subgoals from its past experience based on a combination of goal similarity, novelty, and reachability, and if conducting the task requires taking steps that are very different from the past agent's experience, DISCOVER will never suggest those as subgoals, stopping the learning process. PIM implicitly uses goal-preimage states as subgoals, but it doesn't explicitly pick them from its past experience so it can explore truly new skills. Moreover, Figure 13 shows that PIM's exploration goes down substantially after reaching the goal, whereas DISCOVER continues exploring at roughly the same rate, preventing it from locking in success.

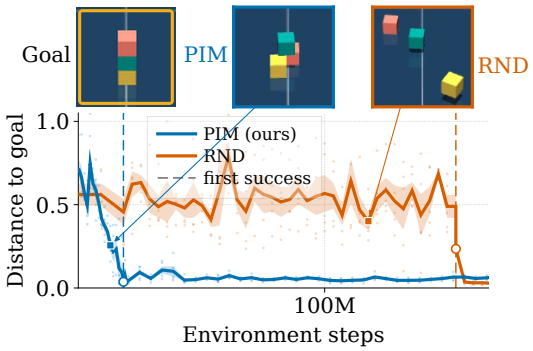


Figure 5: PIM reward guides the policy towards the goal before it is reached. The distance between the visited states and the goal starts to decrease many time steps before the first success. In contrast, RND shows goal-undirected, pure novelty seeking: the policy explores randomly until it stumbles upon the goal.

Figure 6 shows how PIM constructs a 7-cube zig-zag tower and learns to build shorter towers as scaffolding for taller towers during training. Appendix Figure 12 tracks the NF goal latent $z = f_\phi(g; s_0, a_0)$, revealing the agent's evolving belief about the goal. The latent moves from a low- to high-density region, becoming closest to the density peak near the first success, indicating temporary overconfidence in goal reachability. As success stabilizes, it settles in a moderate-density region.



Figure 6: **Progression toward the goal during training.** At iter 1.2k, the purple cube is the highest (as in the goal) but it is too high, and the orange cube is the rightmost but too far to the right. At iters 1.4k and 1.6k, the agent matches some parts of the goal, but has not yet learned to use the green cube as scaffolding for the yellow cube, so the structure is unstable. By iter 3.6k, the agent learns the scaffolding.

5.3 Ablating the state density estimator (Q3)

We use several other density estimators to estimate $\log p^\pi_\gamma(g \mid s, a)$ and use this as the PPO reward. The goal is to test whether the success of PIM is tied to a specific density estimator. For a fair

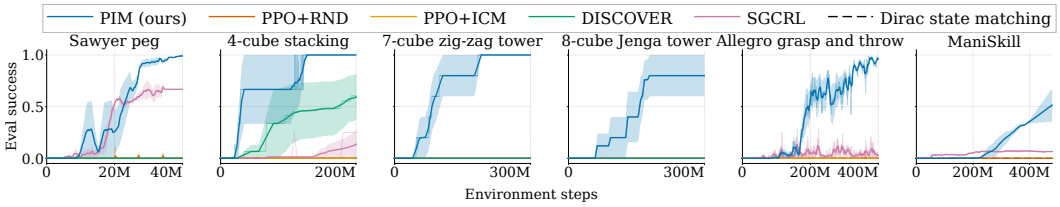


Figure 7: **PIM outperforms the state-of-the-art exploration baselines in all tasks,** with most baselines failing to achieve a single successful episode on many of the tasks. RND (Burda et al., 2019) and ICM (Pathak et al., 2017) use intrinsic novelty seeking bonus, DISCOVER (Diaz-Bone et al., 2025) uses sub-goal selection. SGCRL (Liu et al., 2025) trains policies for all goals. **PIM avoids all of these complexities**. See more tasks in Fig. 10. Mean ± std err over 3 seeds.

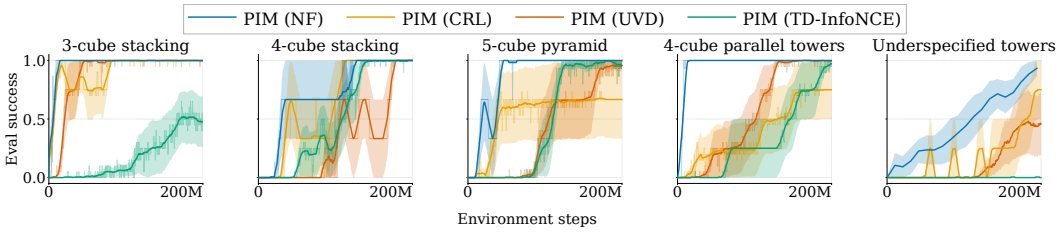


Figure 8: **Underlying generative model.** PIM instantiated with CRL, UVD, and TD-InfoNCE density estimators induce exploration in long-horizon setups, but normalizing flows outperform others. Mean ± std err over 3 seeds.

comparison, we use the same six-layer MLP architecture with residual connections and LayerNorm as in (Wang et al., 2025) (Details in App. C.5) for all density estimators (Except for NF, which has to have a certain invertible architecture). (i) We use CRL, which estimates $\log \frac{p^\pi_\gamma(g|s,a)}{p^\pi_\gamma(g)}$ (Eysenbach et al., 2022). Since the denominator is independent of s and a, using this as reward yields the same optimal policy as $r(s,a) = \log p^\pi_\gamma(g \mid s,a)$. (ii) We use Universal Value Density Estimation (UVD) which uses a TD objective to estimate a smoothed goal reachability probability. Finally, we use TD-InfoNCE (Zheng et al., 2024), which can be viewed as a TD-based version of contrastive learning. This allows us to compare MC and TD training while keeping the underlying contrastive objective similar (App. C.4). Overall, our results in Fig. 8 show that, first, a variety of density estimators are capable of solving long-horizon tasks; second, NF has the best performance and CRL has the second-highest performance. CRL is capable of solving a 7-cube task (Fig 11) that TD-based losses aren't capable of solving. Overall, TD losses are weaker in exploration; however, they are better at maintaining consistent success after succeeding once without any regularization, and their log-likelihood estimates are smoother and monotonic (See 5.4 and C.4.1). In contrast, NF requires additional regularization to retain success (discussion in 5.4). We also tried a flow-matching (FM) model, but it failed on the challenging tasks. It remains unclear whether FM is inherently unable to guide exploration or whether additional engineering is needed to make it work. In App. C.9, we use a didactic experiment to study how the choice of training objective affects out-of-distribution generalization. This is important for our method because exploration is driven by the density estimator's predictions on unseen state g. Our experiment shows that, under the same parameterization and a static dataset, TD, maximum-likelihood, and CRL losses have similar asymptotic generalization properties. Therefore their performance differences in our online exploration setting are due to the dynamic data-collection process.

5.4 Ensuring the log-likelihood is a suitable reward function (Q4)

We observed that on the harder tasks, PIM reaches the goal but soon afterwards loses success. We found that right after the first success, states closer to the goal have higher log-likelihood rewards (intended behavior), but after several updates, the reward decrease along the successful trajectories (Fig. 9). Moreover the rate of change of reward with changing the state explodes after success. Even slightly moving the blocks ($t = 29$ to 68 in Fig. 9) results in a drastic change of reward. We hypothesize this is due to overfitting. After the first success, the collected trajectories become highly concentrated around the goal, resulting in the NF overfitting to limited data. Monitoring the NF training and validation loss before and after success proves this hypothesis; we

observe that while training loss goes from 1.6 down to -4.2 after success, the validation loss significantly increases from 7 to 18. To mitigate this issue, we introduce a state-smoothness regularizer; Prior works (Giammarino et al., 2025) have also observed that trained rewards can be non-smooth and non-monotonic without regularization. Our regularization constrains $\|\nabla_s \log p_\phi(s_f \mid s, a)\|_2$ to remain below 100. We found this threshold to work best empirically. After applying the regularization, we observe that states closer to the goal consistently receive higher reward (as intended) and success becomes stable (more details in App. B.6 and C.6). Finally, we ask: does matching geometrically discounted goal predecessor states provide an advantage over matching the immediate predecessor state? To test this, we train the NF to model one-step dynamics and use $\log p(g \mid s, a)$ as the reward. This variant fails on all of the more challenging tasks.

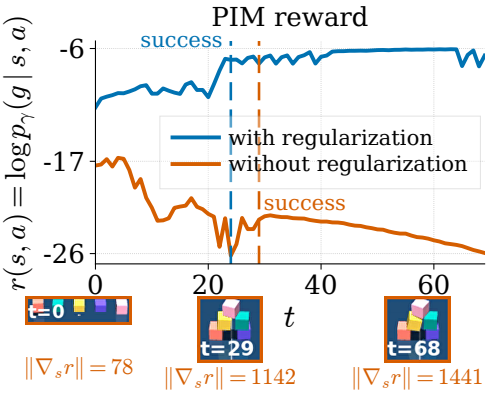


Figure 9: Without any regularization, a few iterations after the first success, NF reward is lower for states closer to the goal, and reward is drastically different for states close to each other, resulting in unstable behavior (orange). Regularizing the reward gradient fixes this (blue).

5.5 Comparing PIM with goal distribution state matching (Q5)

Prior approaches to distribution matching for goal reaching directly match the state distribution to the goal distribution $\delta_g(s_f)$ (Durugkar et al., 2021; Agarwal et al., 2023). In App. C.10, we show that this objective provides no goal-directed signal before the goal is reached and instead behaves like state-entropy maximization, inheriting the same weakness as novelty-based exploration. We further show that it does not automatically stop exploring after finding the goal, and that its objective is highly non-smooth upon observing the goal, making it unsuitable for gradient-based optimization.

6 Conclusion and Future Work

Recent major advances in supervised learning have largely been driven by using massive datasets to train generative models and taking advantage of their generalization abilities (Kaplan et al., 2020; Brown et al., 2020). Similar progress has remained elusive in RL because RL agents should learn to collect their own data. In this work, we show that the generalization abilities of generative models that are trained from scratch are sufficient to lead data collection and solve long-horizon problems. Our work uncovers new ways of using the generalization capabilities of generative modeling in RL, an area that has largely remained underexplored.

Limitations. Our work has several limitations. First, future work could investigate whether other generative models, such as flow-matching or diffusion models, can similarly guide exploration. Second, we do not study how the architecture of the density estimator affects its generalization. Finally, we train the density estimator from scratch; it remains unclear whether initializing from a pretrained density estimator and continually updating it with newly collected data would further improve exploration.

7 STATEMENTS

AI USE STATEMENT

In this work, we used generative AI tools for grammar and spelling checks, word choice, rephrasing, and assistance with editing code and launching experiments.

We did NOT use generative AI tools to develop theoretical results, write full paragraphs or pages, identify citations, or generate research ideas. Any AI-generated code was reviewed by at least one author. The authors take full responsibility for the final content of this work, including the text, claims, code, and experimental results.

ETHICS STATEMENT

Our work develops an online RL algorithm that explores by using the generalization capabilities of generative models and therefore has no immediate ethical concerns.

REPRODUCIBILITY STATEMENT

We will provide the source code for all of our experiments upon publication. The hyperparameters used for main experiments are given in Appendix C. The proofs of all theoretical results are provided in Appendix B.

REFERENCES

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*, 2018.

Jacob Adamczyk, Volodymyr Makarenko, Stas Tiomkin, and Rahul V. Kulkarni. Bootstrapped reward shaping. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 15302–15310, 2025. doi: 10.1609/aaai.v39i15.33679.

Siddhant Agarwal, Ishan Durugkar, Peter Stone, and Amy Zhang. f-policy gradients: A general framework for goal-conditioned rl using f-divergences. In *Advances in Neural Information Processing Systems*, volume 36, pp. 12100–12123, 2023.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, volume 30, pp. 5048–5058, 2017.

Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. Minimax regret bounds for reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 263–272. PMLR, 2017.

Mahsa Bastankhah, Grace Liu, Dilip Arumugam, Thomas L. Griffiths, and Benjamin Eysenbach. Demystifying the mechanisms behind emergent exploration in goal-conditioned rl. In *International Conference on Learning Representations*, 2026.

Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Rémi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, volume 29, pp. 1471–1479, 2016.

Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 449–458. PMLR, 2017.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin,

Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019.

Felipe Codevilla, Matthias Müller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

Leander Diaz-Bone, Marco Bagatella, Jonas Hübotter, and Andreas Krause. Discover: Automated curricula for sparse-reward reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-0738.

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. In *International Conference on Learning Representations Workshop Track*, 2015.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. In *International Conference on Learning Representations*, 2017.

Ishan Durugkar, Mauricio Tec, Scott Niekum, and Peter Stone. Adversarial intrinsic motivation for reinforcement learning. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA, 2021. Curran Associates Inc. ISBN 9781713845393.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. First return, then explore. *Nature*, 590:580–586, 2021. doi: 10.1038/s41586-020-03157-9.

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Ruslan Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 35603–35620, 2022.

Homayoon Farrahi and A. Rupam Mahmood. Learning without time-based embodiment resets in soft-actor critic. In Sarath Chandar, Razvan Pascanu, Eric Eaton, Bing Liu, Rupam Mahmood, and Amal Rannen-Triki (eds.), *Proceedings of The 4th Conference on Lifelong Learning Agents*, volume 330 of *Proceedings of Machine Learning Research*, pp. 112–135. PMLR, 2026.

Chelsea Finn, Paul F. Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *CoRR*, abs/1611.03852, 2016.

Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1515–1528. PMLR, 2018.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1587–1596. PMLR, 2018.

Seyed Kamyar Seyed Ghasemipour, Richard Zemel, and Shixiang Gu. A divergence minimization perspective on imitation learning methods. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura (eds.), *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1259–1277. PMLR, 2020.

Raj Ghugare and Benjamin Eysenbach. Normalizing flows are capable models for continuous control. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Raj Ghugare, Roger Creus Castanyer, Catherine Ji, Kathryn Wantlin, Jin Schofield, Karthik Narasimhan, and Benjamin Eysenbach. Builderbench: The building blocks of intelligent agents. In *RL Beyond Rewards Workshop at the Reinforcement Learning Conference*, 2026.

Vittorio Giammarino, Ruiqi Ni, and Ahmed H. Qureshi. Physics-informed value learner for offline goal-conditioned reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, Xiaodi Yuan, Pengwei Xie, Zhiao Huang, Rui Chen, and Hao Su. Maniskill2: A unified benchmark for generalizable manipulation skills. In *International Conference on Learning Representations*, 2023.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.

Ankur Handa, Arthur Allshire, Viktor Makoviychuk, Aleksei Petrenko, Ritvik Singh, Jingzhou Liu, Denys Makoviichuk, Karl Van Wyk, Alexander Zhurkevich, Balakumar Sundaralingam, Yashraj Narang, et al. Dextreme: Transfer of agile in-hand manipulation from simulation to reality. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5977–5984. IEEE, 2023.

Elad Hazan, Sham Kakade, Karan Singh, and Abby Van Soest. Provably efficient maximum entropy exploration. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2681–2691. PMLR, 2019.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pp. 4572–4580, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, volume 29, pp. 1109–1117, 2016.

Edward S. Hu, Richard Chang, Oleh Rybkin, and Dinesh Jayaraman. Planning goals for exploration. In *International Conference on Learning Representations*, 2023.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G. M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022.

Chi Jin, Zeyuan Allen-Zhu, Sébastien Bubeck, and Michael I. Jordan. Is q-learning provably efficient? In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, pp. 267–274, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc. ISBN 1558608737.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Hilbert J. Kappen, Vicenç Gómez, and Manfred Opper. Optimal control as a graphical model inference problem. *Machine Learning*, 87(2):159–182, 2012.

J. Zico Kolter and Andrew Y. Ng. Near-bayesian exploration in polynomial time. In *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009.

Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric Xing, Sergey Levine, and Ruslan Salakhutdinov. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*, 2019.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

Grace Liu, Michael Tang, and Benjamin Eysenbach. A single goal is all you need: Skills and exploration emerge from contrastive rl without rewards, demonstrations, or subgoals. In *International Conference on Learning Representations*, 2025.

Haozhe Ma, Zhengding Luo, Thanh Vinh Vo, Kuankuan Sima, and Tze-Yun Leong. Highly efficient self-adaptive reward shaping for reinforcement learning. In *International Conference on Learning Representations*, 2025.

Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training. In *International Conference on Learning Representations*, 2023.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

Ajay Mandlekar, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, Albert Tung, Julian Gao, John Emmons, Anchit Gupta, Emre Orbay, Silvio Savarese, and Li Fei-Fei. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Proceedings of the 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pp. 879–893. PMLR, 2018.

Zakaria Mhammedi and James Cohan. Decoupling exploration and policy optimization: Uncertainty guided tree search for hard exploration. *CoRR*, abs/2603.22273, 2026.

Faisal Mohamed, Catherine Ji, Benjamin Eysenbach, and Glen Berseth. Temporal representations for exploration: Learning complex exploratory behavior without extrinsic rewards. In *The Fourteenth International Conference on Learning Representations*, 2026. URL `https://openreview.net/forum?id=KjYpHySlb0`.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3M: A universal visual representation for robot manipulation. In *Proceedings of the 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 892–909. PMLR, 2023.

Ian Osband, Daniel Russo, and Benjamin Van Roy. (more) efficient reinforcement learning via posterior sampling. In *Advances in Neural Information Processing Systems*, volume 26, 2013.

Malayandi Palan, Gleb Shevchuk, Nicholas Charles Landolfi, and Dorsa Sadigh. Learning reward functions by integrating human demonstrations and preferences. In *Proceedings of Robotics: Science and Systems*, Freiburg im Breisgau, Germany, June 2019. doi: 10.15607/RSS.2019.XV.023.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2778–2787. PMLR, 2017.

Aleksei Petrenko, Arthur Allshire, Gavriel State, Ankur Handa, and Viktor Makoviychuk. Dexpbt: Scaling up dexterous manipulation for hand-arm systems with population based training. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.037.

Silviu Pitis, Harris Chan, Stephen Zhao, Bradly Stadie, and Jimmy Ba. Maximum entropy gain exploration for long horizon multi-goal reinforcement learning. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7750–7761. PMLR, 2020.

Vitchyr Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7783–7792. PMLR, 2020.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with CLIP latents. *arXiv preprint arXiv:2204.06125*, 2022.

Tim G. J. Rudner, Vitchyr H. Pong, Rowan McAllister, Yarin Gal, and Sergey Levine. Outcome-driven reinforcement learning via variational inference. In *Advances in Neural Information Processing Systems*, volume 34, pp. 13045–13058, 2021.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Yannick Schroecker and Charles Isbell. Universal value density estimation for imitation learning and goal-conditioned reinforcement learning. *arXiv preprint arXiv:2002.06473*, 2020.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1889–1897. PMLR, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Younggyo Seo, Lili Chen, Jinwoo Shin, Honglak Lee, Pieter Abbeel, and Kimin Lee. State entropy maximization with random encoders for efficient exploration. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9443–9454. PMLR, 2021.

Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1134–1141. IEEE, 2018. doi: 10.1109/ICRA.2018.8462891.

Harshit Sikchi, Caleb Chuck, Amy Zhang, and Scott Niekum. A dual approach to imitation learning from observations with offline datasets. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 1125–1147. PMLR, 06–09 Nov 2025.

Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 12, pp. 1057–1063, 1999a.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1–2):181–211, 1999b. doi: 10.1016/S0004-3702(99)00052-1.

Jean Tarbouriech, Tor Lattimore, and Brendan O'Donoghue. Probabilistic inference in reinforcement learning done right. In *Advances in Neural Information Processing Systems*, volume 36, pp. 33687–33725, 2023.

Alexander Trott, Stephan Zheng, Caiming Xiong, and Richard Socher. Keeping your distance: Solving sparse reward tasks using self-balancing shaped rewards. In *Advances in Neural Information Processing Systems*, volume 32, pp. 10376–10386, 2019.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3540–3549. PMLR, 2017.

Kevin Wang, Ishaan Javali, Michał Bortkiewicz, Tomasz Trzcinski, and Benjamin Eysenbach. 1000 layer networks for self-supervised RL: Scaling depth can enable new goal-reaching capabilities. In *Advances in Neural Information Processing Systems*, volume 38, pp. 175040–175067, 2025. doi: 10.52202/085713-5268.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *International Conference on Learning Representations*, 2023.

David Yan. Builderbench. `https://github.com/David-Yan1/builderbench`, 2026. GitHub repository.

Shih-Min Yang, Martin Magnusson, Johannes A. Stork, and Todor Stoyanov. KEA: Keeping exploration alive by proactively coordinating exploration strategies. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 70927–70942. PMLR, 2025.

Weirui Ye, Yunsheng Zhang, Haoyang Weng, Xianfan Gu, Shengjie Wang, Tong Zhang, Mengchen Wang, Pieter Abbeel, and Yang Gao. Reinforcement learning with foundation priors: Let embodied agent efficiently learn on its own. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 185–208. PMLR, 2025.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura (eds.), *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1094–1100. PMLR, 2020.

Tianjun Zhang, Huazhe Xu, Xiaolong Wang, Yi Wu, Kurt Keutzer, Joseph E. Gonzalez, and Yuandong Tian. Bebold: Exploration beyond the boundary of explored regions. In *International Conference on Learning Representations*, 2021.

Chongyi Zheng, Ruslan Salakhutdinov, and Benjamin Eysenbach. Contrastive difference predictive coding. In *International Conference on Learning Representations*, 2024.

Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *AAAI Conference on Artificial Intelligence*, pp. 1433–1438, 2008.

A Related work

Reward Shaping When prior knowledge or expert feedback are available, prior works have used manually designed rewards (Palan et al., 2019; Trott et al., 2019). When demonstrations of successful trajectories are available prior works train a Q-function and use it as a shaped reward Ma et al. (2025); Adamczyk et al. (2025); Rudner et al. (2021); Sermanet et al. (2018); Ma et al. (2023; 2025); Adamczyk et al. (2025). We similarly use a trained density estimator and its log-likelihood as a shaped reward, but our method doesn't require any successful demonstrations. Instead, the

shaped reward evolves online during training, and, surprisingly, can drive exploration even before the agent has ever succeeded. Rudner et al. (2021) also uses a trained one step goal log likelihood term $\log p(s' = g \mid s, a)$ as reward and they train the state predictor model using online data similar to us. However their method is not equivalent to matching the geometrically discounted state distributions preceding the goal and is not able to solve long-horizon tasks.

Goal-Conditioned RL Prior approaches explore using a distribution of subgoals selected based on novelty, goal similarity, or state-space coverage (Hu et al., 2023; Pitis et al., 2020; Diaz-Bone et al., 2025). Florensa et al. (2018); Pong et al. (2020) use generative models to sample suitable subgoals for exploration. In contrast, our method requires no subgoal selection and matches the final goal-pre-image distribution from the beginning. Liu et al. (2025); Bastankhah et al. (2026) observe similar emergent exploration behavior for hard goals, but their method is different from ours in two important ways. First, they use an off-policy actor objective that train policies for *all goals* and this is necessary for their algorithm to work (Liu et al., 2025). In contrast, we show that a *single PPO policy* is enough. Second, and most importantly, they attribute the exploration behavior to the specific evolution of Contrastive RL (CRL) representations (Bastankhah et al., 2026), whereas we show that exploration is universal across different density estimators, with normalizing flows (NF) even outperforming CRL on several tasks.

B Theoretical derivations and proofs

B.1 Negative binomial time discounting

For discounting future timesteps the geometric distribution is a special case of the generalized negative-binomial distribution.

Let $T \sim \mathrm{NegBin}(r, p),\ r > 0,\ p \in (0, 1)$, (the number of times we fail until we see exactly r success, with success occurring w.p p) with probability mass function (PMF)

$$\Pr(T = t) = \frac{\Gamma(t + r)}{\Gamma(r)\Gamma(t + 1)}\, p^r (1 - p)^t, \qquad t = 0, 1, 2, \ldots. \tag{9}$$

When $r = 1$, this reduces to the geometric distribution used in the standard discounted occupancy measure (Eq. 1). For $r > 1$, the weighting places more mass on later timesteps compared with the geometric distribution, and for $r < 1$, the weighting is more concentrated on earlier timesteps. We therefore define the generalized discounted occupancy measure $p^\pi_{\gamma,r}(s_f \mid s, a)$ as

$$p^\pi_{\gamma,r}(s_f \mid s, a) \coloneqq \overset{\pi}{\Pr}\left(S_T = s_f \mid S_0 = s, A_0 = a\right), \qquad T \sim \mathrm{NegBin}(r, 1 - \gamma) \tag{10}$$

$$= \sum_{t=0}^{\infty} \frac{\Gamma(t + r)}{\Gamma(r)\Gamma(t + 1)}\, (1 - \gamma)^r \gamma^t\, \overset{\pi}{\Pr}\left(S_t = s_f \mid S_0 = s, A_0 = a\right). \tag{11}$$

Whenever r is omitted, i.e., when we write p^π_γ, we refer to the standard geometric case $r = 1$. For integer r, this simplifies to:

$$p^\pi_{\gamma,r}(s_f \mid s, a) = \sum_{t=0}^{\infty} \frac{(t + r - 1)!}{(r - 1)!(t)!}\, (1 - \gamma)^r \gamma^t\, \overset{\pi}{\Pr}\left(S_t = s_f \mid S_0 = s, A_0 = a\right). \tag{12}$$

B.2 VARIATIONAL LOWER BOUND

We start obtaining the ELBO by simplifying the integral $\int p^\pi_\gamma(g \mid s)p^\pi_\gamma(s)\,ds$:

$$\begin{aligned}
\int p^\pi_\gamma(g \mid s)p^\pi_\gamma(s)\,ds &= (1-\gamma)^2 \int \sum_t \gamma^t \overset{\pi}{\Pr}(S_t = g \mid S_0 = s) \cdot \sum_\tau \gamma^\tau \overset{\pi}{\Pr}(S_\tau = s \mid S_0 = s_0)\,ds \\
&= (1-\gamma)^2 \int \sum_{t,\tau} \gamma^{t+\tau} \overset{\pi}{\Pr}(S_t = g \mid S_0 = s) \overset{\pi}{\Pr}(S_\tau = s \mid_0 = s_0)\,ds \\
&= (1-\gamma)^2 \int \sum_{t,\tau} \gamma^{t+\tau} \overset{\pi}{\Pr}(S_{t+\tau} = g \mid S_\tau = s) \overset{\pi}{\Pr}(S_\tau = s \mid S_0 = s_0)\,ds \\
&= (1-\gamma)^2 \int \sum_{t,\tau} \gamma^{t+\tau} \overset{\pi}{\Pr}(S_\tau = s, S_{t+\tau} = g \mid S_0 = s_0)\,ds \\
&= (1-\gamma)^2 \sum_{t,\tau} \gamma^{t+\tau} \int \overset{\pi}{\Pr}(S_\tau = s, S_{t+\tau} = g \mid S_0 = s_0)\,ds \\
&= (1-\gamma)^2 \sum_{t,\tau} \gamma^{t+\tau} \overset{\pi}{\Pr}(S_{t+\tau} = g \mid S_0 = s_0) \\
&= (1-\gamma)^2 \sum_{t=0}^{\infty} \sum_{t'=t}^{\infty} \gamma^{t'} \overset{\pi}{\Pr}(S_{t'} = g \mid S_0 = s_0) \\
&= (1-\gamma)^2 \sum_{t'=0}^{\infty} \sum_{t=0}^{t'} \gamma^{t'} \overset{\pi}{\Pr}(S_{t'} = g \mid S_0 = s_0) \\
&= (1-\gamma)^2 \sum_{t'=0}^{\infty} (t'+1)\,\gamma^{t'} \overset{\pi}{\Pr}(S_{t'} = g \mid S_0 = s_0) \\
&= p^\pi_{\gamma,2}(g \mid s_0) \qquad (13)
\end{aligned}$$

Where we used the Markovian property and the change of variable $t' \coloneqq t + \tau$. $p^\pi_{\gamma,2}(g \mid s_0)$ is the discounted goal reachability with the discount following a negative binomial distribution with $r = 2$, refer to App. B.1 for the definition of $p^\pi_{\gamma,2}(g \mid s_0)$. In the main text, we refer to this by $p^\pi_{\tilde{\gamma}}(g \mid s_0)$ or $p^\pi_{\tilde{\gamma}}(g \mid s_0)$. Note that, if we alternatively use discounting with $r = \frac{1}{2}$ for the inner parts of the integral i.e., $\int p^\pi_{\gamma,\frac{1}{2}}(g \mid s)p^\pi_{\gamma,\frac{1}{2}}(s)\,ds$ then the resulting outcome would have been $p^\pi_\gamma(g \mid s_0) = \int p^\pi_{\gamma,\frac{1}{2}}(g \mid s)p^\pi_{\gamma,\frac{1}{2}}(s)\,ds$. This is because sum of two independent negative binomial random variable with parameters r_1 and r_2 and same success probability $(1-\gamma)$, is a negative binomial random variable with $r = r_1 + r_2$ and same success probability. Therefore, we can get desirable discounting in the objective $p^\pi_\gamma(g \mid s_0)$ by choosing the right discounting inside the integral. Now we are ready to obtain ELBO:

$$\begin{aligned}
\log p^\pi_{\tilde{\gamma}}(g) &\overset{13}{=} \log \int_s p^\pi_\gamma(g \mid s)p^\pi_\gamma(s)\,ds \\
&= \log \int_s p^\pi_\gamma(g \mid s)p^\pi_\gamma(s)\frac{q(s)}{q(s)}\,ds \\
&\overset{\text{Jensen}}{\geq} \int_s q(s)\Big(\log p^\pi_\gamma(g \mid s) + \log p^\pi_\gamma(s) - \log q(s)\Big)\,ds \\
&= \mathbb{E}_{q(s)}[\log p^\pi_\gamma(g \mid s)] - D_{KL}\big(q(s) \| p^\pi_\gamma(s)\big)
\end{aligned}$$

We can use calculus of variations with a Lagrange multiplier to obtain the optimal $q^*(s)$. Define

$$\begin{aligned}
\mathcal{L}(q,\lambda) = \int q(s)\left[\log p^\pi_\gamma(g \mid s) + \log p^\pi_\gamma(s) - \log q(s)\right] ds \\
+ \lambda\left(\int q(s)\,ds - 1\right)
\end{aligned}$$

$$\frac{\delta \mathcal{L}}{\delta q(s)} = \log p_\gamma^\pi(g \mid s) + \log p_\gamma^\pi(s) - \log q(s) - 1 + \lambda$$

Setting the derivative to zero,

$$\log q^*(s) = \log p_\gamma^\pi(g \mid s) + \log p_\gamma^\pi(s) + \lambda - 1$$

Hence,

$$q^*(s) \propto p_\gamma^\pi(g \mid s) p_\gamma^\pi(s)$$

Normalizing $q^*(s)$ yields

$$q^*(s) = \frac{p_\gamma^\pi(g \mid s) p_\gamma^\pi(s)}{\int p_\gamma^\pi(g \mid s') p_\gamma^\pi(s')\, ds'}$$

Finally, the only thing that is left is to show that when using the ELBO derivation for policy optimization, we can safely ignore the dependency of the first term, i.e., $\mathbb{E}_{q(s)}[\log p_\gamma^\pi(g \mid s)]$ on π and only use the second term, i.e., $-D_{KL}(q(s) \| p_\gamma^\pi(s))$. Our observation is that $\max_\pi \mathbb{E}_{q(s)}[\log p_\gamma^\pi(g \mid s)]$ is equivalent to a reward maximization objective with sparse reward $r(s) = \mathbb{1}[s = g]$, where the initial distribution $\rho(s)$ is replaced with $q(s)$. Whereas, as we will show in App B.3, the term $-D_{KL}(q(s) \| p_\gamma^\pi(s))$ can be approximated up to first order with a dense reward maximization objective with the dense (experiments show this reward is dense in practice due to generative model generalization even before seeing the goal) reward $r(s) = \log p_\gamma^{\pi_{\text{old}}}(g \mid s)$, therefore, when optimizing the policy, we can ignore the first term, as this term doesn't produce any non-zero gradient before seeing the goal.

B.3 Policy Gradient

We start by obtaining the gradient of our policy objective to show its equivalence to the policy gradients of a reward maximization problem:

$$\begin{aligned}
\nabla_\theta J(\pi_\theta) &\coloneqq \nabla_\theta D_f\left(p_\gamma^{\pi_\theta}(s) \,\middle\|\, p_\gamma^{\pi_{\text{old}}}(s \mid s_f = g)\right) \\
&= \nabla_\theta \int p_\gamma^{\pi_{\text{old}}}(s \mid s_f = g)\, f\Big(\frac{p_\gamma^{\pi_\theta}(s)}{p_\gamma^{\pi_{\text{old}}}(s \mid s_f = g)}\Big)\, ds \\
&= \int \cancel{p_\gamma^{\pi_{\text{old}}}(s \mid s_f = g)} \frac{\nabla_\theta p_\gamma^{\pi_\theta}(s)}{\cancel{p_\gamma^{\pi_{\text{old}}}(s \mid s_f = g)}}\, f'\Big(\frac{p_\gamma^{\pi_\theta}(s)}{p_\gamma^{\pi_{\text{old}}}(s \mid s_f = g)}\Big)\, ds
\end{aligned} \tag{14}$$

We need to omit the dependency to θ, so we note that when optimizing a policy π_{old}, we obtain the gradient at θ_{old}, therefore we need to evaluate $\nabla_\theta J(\pi_\theta)|_{\theta=\theta_{\text{old}}}$, this is an standard approach to obtain surrogate policy optimization objectives used in prior works as well (Kakade & Langford, 2002; Schulman et al., 2015):

$$\begin{aligned}
\nabla_\theta J(\pi_\theta)|_{\theta=\theta_{\text{old}}} &\overset{14}{=} \int \nabla_\theta p_\gamma^{\pi_\theta}(s)\, f'\Big(\frac{p_\gamma^{\pi_{\text{old}}}(s)}{p_\gamma^{\pi_{\text{old}}}(s \mid s_f = g)}\Big)\, ds \\
&\overset{6}{=} \int \nabla_\theta p_\gamma^{\pi_\theta}(s)\; f'\Big(\frac{Z \cdot \cancel{p_\gamma^{\pi_{\text{old}}}(s)}}{\cancel{p_\gamma^{\pi_{\text{old}}}(s)} \cdot p_\gamma^{\pi_{\text{old}}}(g \mid s)}\Big)\, ds
\end{aligned} \tag{15}$$

And now we compare it with the policy gradient of a reward maximization objective with reward $r(s) = -f'\big(\frac{Z}{p_\gamma^{\pi_{\text{old}}}(g|s)}\big)$:

This means that:

$$\nabla_\theta V^{\pi_\theta}(s_0)|_{\theta_{\text{old}}} = \frac{1}{1-\gamma}\, \nabla_\theta J(\pi_\theta)|_{\theta=\theta_{\text{old}}} \tag{16}$$

the reward obtained from Eq. 15 for the choice of forward KL is calculated as follows:

$$\text{Forward KL} \implies f(x) = x \log x$$

$$f'(x) = \log x + 1 \implies r(s) = \log p_\gamma^{\pi_{\text{old}}}(g \mid s) - \log Z + 1$$

For the sake of reward maximization since $-\log Z + 1$ is constant with respect to state and action we can simply use $r(s) = \log p_\gamma^{\pi_{\text{old}}}(g \mid s)$.

This analysis shows that, our goal preimage matching objective can be approximated with a reward maximization objective up to first order derivatives. Hazan et al. (2019) shows that since f-divergence is a convex function optimizing, if we obtain a mixture of policies and in every policy update add the new policy to the mixture, applying the the corresponding reward maximization problem iteratively does converge to the optimal policy i.e., the policy that minimizes $D_f\left(J(\pi_\theta) = p_\gamma^{\pi_\theta}(s) \,\middle\|\, p_\gamma^{\pi_{\text{old}}}(s \mid s_f = g)\right)$. However, note that this is by considering π_{old} to be fixed, meaning that the result obtained by (Hazan et al., 2019) only shows that for a fixed convex optimization problem w.r.t the occupancy measure, solving the surrogate reward maximization problem is sufficient; however in our case $J(\pi_\theta)$ itself is changing by changing the preimage distribution $p_\gamma^{\pi_{\text{old}}}(s \mid s_f = g)$. So the result we obtained here doesn't show convergence to the optimal goal-reaching policy. In App. B.4 we obtain the policy improvement result, which argues about how preimage distribution changes in every iteration and shows that matching the new preimage distribution is indeed monotonic policy improvement.

B.4 Policy improvement theorem and proof

Algorithm 1 summarizes the ELBO-inspired state-distribution matching algorithm and in this section we show policy improvement guarantees for this EM algorithm. Note that this algorithm only serves as the conceptual and theoretical model behind our algorithm and doesn't describe our practical algorithm that is implemented in our experiments. In practice we used the policy gradient trick to obtain a reward maximization surrogate objective (App. B.3). Our algorithm introduces a new form of iterative policy evaluation and improvement. Rather than estimating a value function and greedily updating the policy with respect to it, our method estimates a goal-preimage distribution and updates the policy to greedily match this state distribution. However, it is not clear that repeatedly applying these updates yields monotonic policy improvement. In this section, we show that, under standard policy-improvement assumptions, the procedure yields monotonic policy improvement and converges to the optimal goal-reaching policy at a faster rate than prior works (Kakade & Langford, 2002) highlighting the superiority of state matching over advantage-based policy improvement. This analysis, however, does not address the exploration properties of the algorithm, as it relies on the standard state-action coverage assumptions used in prior policy-improvement analyses (Kakade & Langford, 2002), but regardless it is a useful results because it shows policy improvement doesn't have to be done through advantage maximization (as the common practice in RL) and can be done through state matching as well and even with a better time complexity.

The following theorem establishes the monotonic policy improvement guarantee of Algorithm 1. The key observation is that, at each iteration, the goal pre-image distribution places more mass on the goal than the occupancy measure of policy at iteration k i.e., π_k. Since the improvement step chooses a policy whose occupancy measure is closest to this pre-image distribution, this additional goal mass must be reflected in the updated policy. Specifically, we show that any candidate policy π_{k+1} satisfying $p_\gamma^{\pi_{k+1}}(g) < C_k\, p_\gamma^{\pi_k}(g), \quad C_k > 1$, cannot be optimal for the f-divergence objective in Equation 18: there exists another feasible policy π' with larger discounted goal-reaching probability that strictly decreases the objective.

The multiplicative improvement factor $C_k > 1$ gives a time complexity independent of $|\mathcal{S} \times \mathcal{A}|$, unlike the classical policy iteration bound. It also improves over conservative policy improvement (Kakade & Langford, 2002), which requires $\mathcal{O}(1/\delta^2)$ steps to reach a policy within δ of optimal return, whereas our algorithm requires only $\mathcal{O}(1/\log(1+(1-\gamma)\delta))$ steps.

Algorithm 1 Goal-preimage state distribution matching algorithm

Input: Initial policy π_0
Output: Converged policy $\pi^\star$

while 1 **do**
  Policy evaluation:
  Compute discounted state occupancy $p_\gamma^{\pi_{\text{old}}}(s)$ of policy π_{old}
  Compute the goal-preimage distribution

$$p_\gamma^{\pi_{\text{old}}}(s \mid s_f = g) = \frac{p_\gamma^{\pi_{\text{old}}}(s)\, p_\gamma^{\pi_{\text{old}}}(g \mid s)}{Z} \tag{17}$$

  Policy improvement:

$$\pi \leftarrow \arg\min_\pi D_f\big(p_\gamma^\pi(s) \,\|\, p_\gamma^{\pi_{\text{old}}}(s \mid s_f = g)\big) \tag{18}$$

  if Policy improvement doesn't change the policy **then**
    return $\pi^\star \leftarrow \pi$
  $\pi_{\text{old}} \leftarrow \pi$

Finally, in contrast to prior RL-as-inference methods such as SAC (Haarnoja et al., 2018) and MPO (Abdolmaleki et al., 2018), which provide guarantees for entropy-regularized and KL-regularized objectives, respectively, our policy improvement theorem establishes improvement of the original, unregularized goal-reaching objective.

Theorem 1. *Consider the suboptimal policy π_k that satisfies $\pi_k(a \mid s) > 0$ for every state–action pair (s, a). Let π_{k+1} be the policy returned by one step of the policy improvement step of Algorithm 1, and let $D_f(\cdot\|c)$ be an f-divergence whose domain contains every distribution supported on $\mathrm{supp}(c)$. Define*

$$p_\gamma^*(g) \coloneqq \max_\pi p_\gamma^\pi(g),$$

Then, there is a constant $C_k > 1$ such that:

$$p_\gamma^{\pi_{k+1}}(g) \geq \min\big\{C_k\, p_\gamma^{\pi_k}(g),\ p_\gamma^*(g)\big\},$$

Equivalently, each policy improvement step of Algorithm 1 either reaches an optimal goal-reaching policy or increases the discounted goal-reaching probability by a multiplicative factor of at least C_k.

Corollary 2. *Assume that, at every iteration, the policy produced by Algorithm 1 assigns positive probability to every action (e.g., by adding an arbitrarily small entropy regularization term). Then, after*

$$\mathcal{O}\left(\frac{1}{\log\big(1 + (1-\gamma)\delta\big)}\right)$$

iterations, the policy π_k satisfies

$$p_\gamma^*(g) - p_\gamma^{\pi_k}(g) \leq \delta.$$

Proof of Theorem 1

Proof. For notational convenience, define the vectors

$$a \coloneqq p_\gamma^{\pi_k}, \qquad b(s) \coloneqq p_\gamma^{\pi_k}(g \mid s),$$

$$c \coloneqq p_{\text{pre}}^k, \implies c(s) = \frac{a(s)b(s)}{\sum_{\bar{s}} a(\bar{s})b(\bar{s})}, \tag{19}$$

and

$$d^* \coloneqq p_\gamma^{\pi_{k+1}}.$$

Convex optimization problem By construction, d^* is the solution of the convex optimization problem

$$\begin{aligned} d^* = \arg\min_d \quad & D_f(d \,\|\, c) \\ \text{s.t.} \quad & \sum_s d(s) = 1, \\ & \sum_a d(s,a) = (1-\gamma)\rho(s) + \gamma \sum_{s',a'} d(s',a') P(s \mid s',a'). \\ & d(s) = \sum_a d(s,a) \end{aligned}$$

Where $\rho(s)$ is the initial state distribution.

Let $\mathcal{D}(c)$ denote the feasible set

$$\mathcal{D}(c) \coloneqq \Big\{ d \in \Delta^{|\mathcal{S}|-1} : \mathrm{supp}(d) \subseteq \mathrm{supp}(c),\ d \text{ satisfies the Bellman flow constraints} \Big\}.$$

Since π_k gives non zero probability to all actions in all states, the states along the optimal trajectories do have a non zero probability under c and therefore the optimal policy occupancy measure is in the feasible set.

First order optimality constraint Since d^* minimizes the convex objective $D_f(\cdot\|c)$ over the convex feasible set $\mathcal{D}(c)$, the first-order optimality condition gives

$$\nabla D_f(d^*\|c)^\top (d - d^*) \geq 0, \qquad \forall d \in \mathcal{D}(c).$$

We will show later how to construct a constant C_k from π_k. The proof proceeds by contradiction, we show that for a "bad" d^* such that $d^*(g) < C_k\, a(g)$, there exists a feasible occupancy measure $d \in \mathcal{D}(c)$ satisfying $\nabla D_f(d^*\|c)^\top (d - d^*) < 0$, proving that, every optimizer must satisfy

$$d^*(g) \geq \min\big\{ C\, a(g),\ p^*_\gamma(g) \big\},$$

thereby proving the theorem.

The gradient of the f-divergence is given component-wise by

$$\nabla D_f(d\|c) = \left[f'\left(\frac{d(s)}{c(s)} \right) \right]_{s \in \mathcal{S}}.$$

It therefore suffices to construct a feasible occupancy measure $d \in \mathcal{D}(c)$ for which the directional derivative w.r.t. the bad optimizer d^* is negative. Specifically, we will show that there exist a state $\hat{s} \neq g$ and a scalar $\varepsilon > 0$ such that

$$d(s) = \begin{cases} d^*(g) + \varepsilon, & s = g, \\ d^*(\hat{s}) - \varepsilon, & s = \hat{s}, \\ d^*(s), & \text{otherwise}, \end{cases}$$

belongs to $\mathcal{D}(c)$ and satisfies

$$\nabla D_f(d^*\|c)^\top (d - d^*) < 0.$$

Note that for this choice of d we have:

$$\nabla D_f(d^*\|c)^\top (d - d^*) = \varepsilon \left(f'\left(\frac{d^*(g)}{c(g)} \right) - f'\left(\frac{d^*(\hat{s})}{c(\hat{s})} \right) \right) \tag{20}$$

So it suffices to show $f'\left(\frac{d^*(g)}{c(g)}\right) - f'\left(\frac{d^*(\hat{s})}{c(\hat{s})}\right) < 0$ and since the derivative of a convex function is increasing, it suffices to show $\frac{d^*(g)}{c(g)} < \frac{d^*(\hat{s})}{c(\hat{s})}$.

Constructing the descent direction Let

$$\mathcal{S}_1 \coloneqq \left\{ s \in \mathcal{S} \setminus \{g\} : d^*(s) = \min_{d \in \mathcal{D}(c)} d(s) \right\}$$

denote the set of states whose occupancy is already at its minimum feasible value, states in $\mathcal{S}_1$ cannot be used to construct the descent direction because their $d^*(s)$ cannot be reduced further. Let

$$\mathcal{S}_2 \coloneqq \mathcal{S} \setminus (\mathcal{S}_1 \cup \{g\})$$

denote the remaining states. By definition, for every $s \in \mathcal{S}_2$, the occupancy $d^*(s)$ can be decreased while remaining in the feasible set.

We next show that there exists a state $\hat{s} \in \mathcal{S}_2$ such that $\frac{d^*(g)}{c(g)} < \frac{d^*(\hat{s})}{c(\hat{s})}$ and we use such $\hat{s}$ to construct the descent direction. Suppose, for the sake of contradiction, that

$$\frac{d^*(g)}{c(g)} \geq \frac{d^*(s)}{c(s)}, \qquad \forall s \in \mathcal{S}_2. \tag{21}$$

We will show that equation 21 implies

$$\sum_{s \in \mathcal{S}} d^*(s) < 1,$$

contradicting the fact that d^* is a probability distribution. Therefore, there exists a state $\hat{s} \in \mathcal{S}_2$ satisfying

$$\frac{d^*(g)}{c(g)} < \frac{d^*(\hat{s})}{c(\hat{s})}.$$

First we note that for any state $s \neq g$, the discounted probability of reaching the goal satisfies

$$b(s) \leq \gamma\, b(g), \tag{22}$$

where the inequality follows since the maximum discounted goal-reaching probability is achieved by transitioning to g in the immediate next step. Then we evaluate $\sum_{s \in S_2} d^*(s)$ and $\sum_{s \in S_1} d^*(s)$:

$$\sum_{s \in S_2} d^*(s) \underset{21}{\leq} \sum_{s \in S_2} \frac{d^*(g) c(s)}{c(g)} \underset{19}{=} \sum_{s \in S_2} \frac{d^*(g) a(s)\, b(s)}{a(g)\, b(g)} \underset{22}{\leq} \frac{\gamma\, d^*(g)}{a(g)} \sum_{s \in S_2} a(s) < \gamma\, C_k \sum_{s \in S_2} a(s) \tag{23}$$

Where the last inequality results from the fact that we assumed d^* is a bad optimizer i.e.,

$$d^*(g) < C_k\, a(g) \tag{24}$$

By the definition of $\mathcal{S}_1$, every state in $\mathcal{S}_1$ already attains its minimum feasible occupancy. Since the occupancy measure of the current policy, a, is feasible, it follows that

$$\sum_{s \in \mathcal{S}_1} d^*(s) \leq \sum_{s \in \mathcal{S}_1} a(s). \tag{25}$$

Combining equation 23, equation 25, and equation 24, we obtain

$$\begin{aligned} \sum_{s \in \mathcal{S}} d^*(s) &= \sum_{s \in \mathcal{S}_1} d^*(s) + \sum_{s \in \mathcal{S}_2} d^*(s) + d^*(g) \\ &< \sum_{s \in \mathcal{S}_1} a(s) + \gamma C_k \sum_{s \in \mathcal{S}_2} a(s) + C a(g) \\ &= \sum_{s \in \mathcal{S}_1} a(s) + C_k \left(\gamma \sum_{s \in \mathcal{S}_2} a(s) + a(g) \right). \end{aligned} \tag{26}$$

Now construct C_k as follows:

$$B \coloneqq \sum_{s \in \mathcal{S}_2} a(s).$$

$$C_k = \frac{B + a(g)}{\gamma B + a(g)},$$

Since π_k is suboptimal, we have $B > 0$ therefore $C_k > 1$. Note that, if $B = 0$, then every state except the goal belongs to $\mathcal{S}_1$ and therefore already attains its minimum feasible occupancy. Consequently, the goal occupancy $a(g)$ is maximal among all feasible occupancy measures, implying that π_k is already optimal and in this case π_{k+1} is already the optimal policy. So if π_{k+1} is not optimal, B is certainly greater than 0 and $C_k > 1$.

Substituting C_k into Equation 26 yields:

$$\sum_{s \in \mathcal{S}} d^*(s) < \sum_{s \in \mathcal{S}_1} a(s) + \sum_{s \in \mathcal{S}_2} a(s) + a(g) = 1,$$

contradicting the fact that d^* is a probability distribution. Therefore, whenever

$$d^*(g) < \min\left\{C_k\, a(g),\ p^*_\gamma(g)\right\},,$$

there exists a state $\hat{s} \in \mathcal{S}_2$ and a sufficiently small $\varepsilon > 0$ such that the perturbed occupancy measure

$$d(s) = \begin{cases} d^*(g) + \varepsilon, & s = g, \\ d^*(\hat{s}) - \varepsilon, & s = \hat{s}, \\ d^*(s), & \text{otherwise}, \end{cases}$$

belongs to the feasible set $\mathcal{D}(c)$ and is a descent direction. Thus, we have constructed a feasible perturbation that transfers occupancy mass from $\hat{s}$ to the goal.

□

Proof of Corollary 2

Proof. Let $k+1$ be the first iteration satisfying

$$p^*_\gamma(g) - p^{\pi_{k+1}}_\gamma(g) \le \delta.$$

We derive an upper bound on k.

By Theorem 1, at iteration i, the goal-reaching probability is multiplied by

$$C_i = \frac{B_i + p^{\pi_i}_\gamma(g)}{\gamma B_i + p^{\pi_i}_\gamma(g)},$$

where

$$B_i = \sum_{s \in \mathcal{S}_2} p^{\pi_i}_\gamma(s),$$

and $\mathcal{S}_2$ is the set of states whose occupancy can still be decreased while maintaining feasibility.

Since both $p^{\pi_i}_\gamma$ and p^*_γ are occupancy distributions,

$$\sum_{s \in \mathcal{S}_1} p^*_\gamma(s) + \sum_{s \in \mathcal{S}_2} p^*_\gamma(s) + p^*_\gamma(g) = \sum_{s \in \mathcal{S}_1} p^{\pi_i}_\gamma(s) + \sum_{s \in \mathcal{S}_2} p^{\pi_i}_\gamma(s) + p^{\pi_i}_\gamma(g).$$

Rearranging gives

$$\begin{aligned} B_i &= \sum_{s \in \mathcal{S}_2} p^{\pi_i}_\gamma(s) \\ &= \left(\sum_{s \in \mathcal{S}_1} p^*_\gamma(s) - \sum_{s \in \mathcal{S}_1} p^{\pi_i}_\gamma(s)\right) + \left(p^*_\gamma(g) - p^{\pi_i}_\gamma(g)\right) + \sum_{s \in \mathcal{S}_2} p^*_\gamma(s). \end{aligned}$$

The first term is nonnegative since every state in $\mathcal{S}_1$ already attains its minimum feasible occupancy under π_i, and the last term is also nonnegative. Therefore,

$$B_i \geq p_\gamma^*(g) - p_\gamma^{\pi_i}(g).$$

Since $i \leq k$,

$$p_\gamma^*(g) - p_\gamma^{\pi_i}(g) > \delta,$$

and hence

$$B_i > \delta.$$

Therefore,

$$\begin{aligned} C_i &= 1 + \frac{(1-\gamma)B_i}{\gamma B_i + p_\gamma^{\pi_i}(g)} \\ &\geq 1 + \frac{(1-\gamma)B_i}{\gamma B_i + 1} \\ &\geq 1 + \frac{(1-\gamma)\delta}{\gamma\delta + 1}, \end{aligned}$$

Hence,

$$p_\gamma^{\pi_{k+1}}(g) \geq \prod_{i=0}^{k} C_i \, p_\gamma^{\pi_0}(g) \geq \left(1 + \frac{(1-\gamma)\delta}{\gamma\delta+1}\right)^{k+1} p_\gamma^{\pi_0}(g).$$

Solving for the smallest k such that

$$p_\gamma^*(g) - p_\gamma^{\pi_{k+1}}(g) \leq \delta$$

yields

$$k = \mathcal{O}\left(\frac{1}{\log(1 + (1-\gamma)\delta)}\right),$$

which completes the proof. □

B.5 CRL-based Goal-preimage matching as approximate Bayesian RL

In this section, we show that when using contrastive RL as occupancy modeling, goal preimage state matching has connection with an approximate form of Bayesian RL methods (Osband et al., 2013; Kolter & Ng, 2009; Tarbouriech et al., 2023). Our analysis in no way aims to show a rigorous mathematical connection between bayes RL methods and CRL and instead is meant to motivate conceptual connections that will hopefully motivate future work.

Our analysis in inspired by the problem formulation presented in (Tarbouriech et al., 2023). They define Bernoulli random variables $\Gamma_\ell^*(s)$ that represents whether state s is on the optimal path at timestep ℓ or not. They argue that one approach to Bayesian RL is to form beliefs about $\Gamma_\ell^*(s)$; and one can infer $\Pr_\alpha[\Gamma_\ell^*(s) = 1]$ (where α is a distribution over the space of all MDPs) using the collected data and a prior over optimal states in a Bayesian way. In Lemma 1 (Tarbouriech et al., 2023) they show that $\Pr_\alpha[\Gamma_\ell^*(s) = 1]$ is a valid occupancy measure and one can attempt to update the policy to match this state distribution i.e., attempting to do $\Pr^\pi[S_t = s] = \Pr[\Gamma_\ell^*(s) = 1]$. Their approach for obtaining such policy is to turn the reward maximization problem into a UCB style reward maximization that takes uncertainty about optimality into account. Whereas here we ask the question: *Can we interpret the preimage distribution as an approximation of* $\Pr_\alpha[\Gamma_\ell^*(s) = 1]$ *and therefore interpret our algorithm as an approximation of a principled bayes RL method?*. Turned out the answer to that question is positive if we use CRL as density estimator and the metric space induced by CRL representations can be interpreted as capturing uncertainty about the goal. Here we present the approximation steps needed to go from a principled Bayes method to the preimage matching with CRL and discuss how the notion of epistemic uncertainty about the goal g will be captured by the similarity to goal in the representation space i.e., $\psi(s)^\top \psi(g)$.

Similar to (Tarbouriech et al., 2023) we define optimality Bernoulli random variables:

$$\Gamma_\ell^*(s) \coloneqq \{S_\ell^* = s\}$$

From lemma 1 of (Tarbouriech et al., 2023), we know that $\mathbb{P}_{\Gamma^*} := \{\mathbb{P}_\alpha(\Gamma^*_\ell)_{\ell=1}^L\}$ is an occupancy measure.

For goal conditioned setups with a single goal, we can define random variable G that denotes the goal. We use a random variable for it to show epistemic uncertainty about the goal, any state could potentially be the goal i.e., for any state s_f we can evaluate $\mathbb{P}_\alpha(G = s_f)$. We can write the probability of optimality of s by marginalizing over G as follows (we drop α everywhere for simplicity in notation):

$$\mathbb{P}[\Gamma^*_\ell(s)] = \sum_{s_f} \mathbb{P}[G = s_f]\, \mathbb{P}[\Gamma^*_\ell(s) \mid G = s_f]$$

$\mathbb{P}[\Gamma^*_\ell(s) \mid G = s_f]$ is the probability of s being optimal at step ℓ if s_f is the goal. Using a policy π with collected trajectories $\{\tau\}$ we can approximate this term with empirical occurrence of s before observing s_f in the collected trajectories:

Approximation 1: We approximate $\mathbb{P}[\Gamma^*_\ell(s) \mid G = s_f] \approx \mathbb{P}^\pi[S_\ell = s \mid S_T = s_f]$ where T is the terminal timestep.

Using this approximation we will have:

$$\begin{aligned}
\mathbb{P}[\Gamma^*_\ell(s)] &= \sum_{s_f} \mathbb{P}[G = s_f]\, \mathbb{P}[\Gamma^*_\ell(s) \mid G = s_f] \\
&= \sum_{s_f} \mathbb{P}[G = s_f]\, \mathbb{P}^\pi[S_\ell = s \mid S_T = s_f] \\
&= \sum_{s_f} \mathbb{P}[G = s_f]\, \frac{\mathbb{P}^\pi[S_T = s_f \mid S_\ell = s]\, \mathbb{P}^\pi[S_\ell = s]}{\mathbb{P}^\pi[S_T = s_f]} \\
&= \mathbb{P}^\pi[S_\ell = s] \cdot \mathbb{E}_{s_f \sim p^\pi(s_T|s)} \left[\frac{\mathbb{P}[G = s_f]}{\mathbb{P}^\pi[S_T = s_f]} \right]
\end{aligned} \tag{27}$$

This simplifies the optimality probability of s as the marginal likelihood of s under the policy and a term that depends on how likely states visited after s are to be goal.

Now we turn our attention to the parameterization of goal preimage distribution if we use CRL as density estimator for the inner product parameterization as denoted in Eq. 36 and we results from (Eysenbach et al., 2022) on $\phi(s,a)^\top \psi(g) = \log \frac{p^\pi(g|s)}{p^\pi(g)}$. Now using this we rewrite the preimage distribution (indexed by timestep).

$$\begin{aligned}
p^\pi(s_\ell \mid s_f = g) &= \frac{p^\pi(g \mid s_\ell) p^\pi(s_\ell)}{Z} \\
&= \frac{\exp\left(\phi(s_\ell)^\top \psi(g)\right) p^\pi_\gamma(s_\ell)}{Z'}
\end{aligned} \tag{28}$$

Now we use the Lemma 2 (Appendix B.2.2) of (Bastankhah et al., 2026). They show that under certain assumptions $\phi(s) = \mathbb{E}_{s_f \sim p^\pi_\gamma(s_f|s)}[\psi(s_f)]$. We plug this into Eq. 28 and get:

$$p^\pi(s_\ell \mid s_f = g) = \frac{\exp\left(\mathbb{E}_{s_f \sim p^\pi_\gamma(s_f|s_\ell)}[\psi(s_f)^\top \psi(g)]\right) p^\pi_\gamma(s)}{Z'} \tag{29}$$

We can use Jensen inequality to write:

$$p^\pi(s_\ell \mid s_f = g) \leq \frac{\mathbb{E}_{s_f \sim p^\pi_\gamma(s_f|s)} \left[\exp\left(\psi(s_f)^\top \psi(g)\right) \right] p^\pi_\gamma(s_\ell)}{Z'} \tag{30}$$

We note that if we consider the Jensen gap $\Delta(s_\ell) := \log \mathbb{E}_{p^\pi_\gamma(s_f|s_\ell)} \Big[\exp\left(\psi(s_f)^\top \psi(g)\right) \Big] - \mathbb{E}_{p^\pi_\gamma(s_f|s_\ell)} \Big[\psi(s_f)^\top \psi(g) \Big]$ since $0 \leq \psi(s_f)^\top \psi(g) \leq 1$ Hoeffding's lemma helps us to bound $\Delta(s) \leq$

$\frac{1}{8}$ and this allows us to bound the difference between CRL goal preimage and the term obtained in 30. This will further allow us to compare the approximate goal preimage distribution of CRL (Eq. 30) with the approximate bayes optimal state distribution$\mathbb{P}[\Gamma^*_\ell(s)]$. A one-to-one comparison of Eq. 27 and Eq. 30 shows that the terms $\mathbb{P}^\pi[S_\ell = s_\ell] = p^\pi(s_\ell)$ are equal, the question is can the remaining terms be equal too? i.e., $\mathbb{E}_{s_f \sim p^\pi(s_T|s)}\left[\frac{\mathbb{P}[G=s_f]}{\mathbb{P}^\pi[S_T=s_f]}\right] = \mathbb{E}_{s_f \sim p^\pi_\gamma(s_f|s)}\left[\exp\left(\psi(s_f)^\top \psi(g)\right)\right]$? This would be true if the following holds:

$$\exp\left(\psi(s_f)^\top \psi(g)\right) = \frac{\mathbb{P}[G = s_f]}{\mathbb{P}^\pi[S_T = s_f]} \tag{31}$$

This means that the inner product of representations $\psi(s_f)^\top \psi(g)$ should capture a notion of epistemic uncertainty about the goal $\mathbb{P}[G = s_f]$. Previous work shows that CRL representations can capture a notion of uncertainty about the goal, i.e., agent's information about whether a state is goal or not (Bastankhah et al., 2026). In particular, theorem 3 (App. B4) of (Bastankhah et al., 2026) shows that upon every GD update on $\psi(s_f)$, its inner product with $\psi(g)$ decreases i.e., $\psi^k(s_f)^\top \psi(g) = \zeta \psi^{k-1}(s_f)^\top \psi(g)$ where $0 < \zeta < 1 - \varepsilon$ for some $\varepsilon > 0$; k denotes the number of times $\psi(s_f)$ has been updated and $\psi^0(s_f)^\top \psi(g) = 1$ (parallel initialization). This means that after representation convergence, $\exp\left(\psi(s_f)^\top \psi(g)\right)$ is 1 for any non-goal state s_f that has been visited and is e for any state that hasn't been visited, higher inner product capturing more optimism about s_f being the goal (higher $\mathbb{P}[G = s_f]$) and lower inner product showing more certainty about the fact that s_f is not the goal.

B.6 A Primer on Normalizing Flows

Normalizing flows (NFs) are generative models that are used to learn an arbitrary probability distribution $p(x)$, where $x \in \mathbb{R}^d$. The main idea is to learn an invertible transformation $f_\phi : \mathbb{R}^d \to \mathbb{R}^d$ that maps samples from the target distribution to samples from a simpler prior distribution $p_0(z)$, where $z = f_\phi(x)$. The parameters of the transformation are learned by maximum likelihood. A key requirement is that f_ϕ is invertible and that both its inverse and Jacobian determinant can be computed efficiently (Papamakarios et al., 2021). Using the change-of-variables formula, the log-likelihood of the resulting density is:

$$\log p_\phi(x) = \log p_0(f_\phi(x)) + \log \left| \det \frac{\partial f_\phi(x)}{\partial x} \right|. \tag{32}$$

for sampling from the learned distribution we first sample from the prior and then apply the inverse transformation: $z \sim p_0(z),\ x = f_\phi^{-1}(z)$. The prior is an Isotropic Gaussian distribution.

In this work, we use a conditional normalizing flow to model the discounted future-state occupancy distribution $p^\pi_\gamma(s_f \mid s, a)$, conditioned on the current state-action pair (s, a) (Ghugare & Eysenbach, 2025). In particular, we parameterize the transformation as $f_\phi(s_f; s, a)$, where (s, a) provides the conditioning information. The model is trained by maximizing the conditional log-likelihood,

$$\max_\phi \ \mathbb{E}_{(s,a),\, s_f \sim p^\pi_\gamma(\cdot|s,a)} \left[\log p_0(f_\phi(s_f; s, a)) + \log \left| \det \frac{\partial f_\phi(s_f; s, a)}{\partial s_f} \right| \right]. \tag{33}$$

Algorithm Summary We present the details of the Normalizing Flows training in Algorithm 2. We train the conditional NF (conditioned on s, a) to learn the distribution of states s_f discounted geometrically from the future of s, a. Following the prior works, NF is trained using the log likelihood loss. More information about the NF architecture can be found in App. C.5. We use a dual regularizer with an update-able dual variable λ that applies the constraint $\mathbb{E}[\|\nabla_s \log p_\phi(s_f; s, a)\|] \leq c$. Moreover we found that using a slightly smoothened version of the NF parameters for obtaining the PPO reward helps with stability so we also strore a moving average version of the Nf parameters. In practice long horizon moving average harms the performance and we found $\eta = 0.5$ is most helpful.

Algorithm 2 Normalizing Flow Training

1: Sample a batch of B trajectories $\{\tau_i\}_{i=1}^{B}$ from the replay buffer.
2: **for** each trajectory τ_i **do**
3: Sample a state-action pair (s_t^i, a_t^i) from τ_i.
4: Sample a future horizon

$$T \sim \text{Geom}(1-\gamma), \qquad T \leftarrow \min(T, H-t),$$

and set $s_f^i \leftarrow s_{t+T}$.
5: Compute the conditional log likelihood

$$\log p_\phi(s_f^i; s_t^i, a_t^i) = -\frac{1}{2}\left\|f_\phi(s_f^i; s_t^i, a_t^i)\right\|^2 - \frac{d}{2}\log(2\pi) + \log\left|\det \frac{\partial f_\phi(s_f^i; s_t^i, a_t^i)}{\partial s_f^i}\right|$$

6: Compute the main loss

$$\mathcal{L}_\phi = -\sum_i \log p_\phi(s_f^i; s_t^i, a_t^i)$$

7: Compute the smoothness regularizer loss:

$$\mathcal{L}_\phi^{\text{reg}} = \sum_i \|\nabla_s \log p_\phi(s_f^i; s_t^i, a_t^i)\|_2 - c$$

8: Update NF parameters and λ by gradient descent/ascent:

$$\phi \leftarrow \phi - \eta\nabla_\phi\left(\mathcal{L}_\phi + \lambda\mathcal{L}_\phi^{\text{reg}}\right).$$

$$\lambda \leftarrow \max\left(0, \lambda + \eta\mathcal{L}_\phi^{\text{reg}}\right)$$

9: Update the exponential moving average parameters:

$$\bar{\phi} \leftarrow (1-\zeta)\bar{\phi} + \zeta\phi.$$

C Experiment Details

C.1 More Experimental Results

In this section, we present extra plots. Figure 10 presents the additional baseline experiments. We observe that while the baselines can solve simpler block building tasks such as stacking 3 cubes, completely fail in harder tasks such as 7 cube zig zag tower that involves more blocks and requires more sophisticated strategies such as using scaffolding to build stable structures. We moreover observe that methods that work based on density estimators such as MPO-CRL and SGCRL, outperform methods that work based on novelty such as RND and Dirac state matching as the task becomes more long horizon and finding success by chance becomes astronomically less likely.

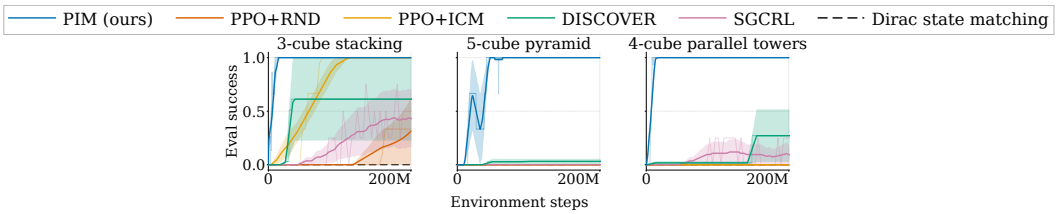


Figure 10: Additional baseline experiments. Our algorithm significantly outperforms baselines in long horizon structure building tasks of BuilderBench.

Figure 11 shows additional results of comparing different density estimators. While NF outperforms the otehr estimators and solves tasks that no other method, other methods show emergent exploration in some hard tasks that baselines struggled with such as 5 cube-pyramid task.

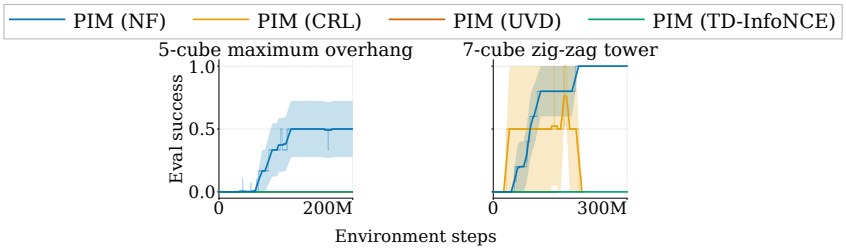


Figure 11: Additional plots for ablating the density estimator. All methods show emergent exploration in hard block building tasks.

Figure 12 shows the evolution of $z = f_\phi(g; s_0, a_0)$ during training where s_0, a_0 are initial state and action. NF maps samples from the data distribution to an isotropic gaussian distribution therefore high density regions lie around the origin. As training starts z starts moving toward the origin showing that the agent is learning to get closer to the goal. z has highest density at first success showing that at first success the agent perceives the goal to be easily reachable. As the agent gets better at constructing the blocks with more details, z goes toward moderate density regions showing that the agent doesn't overestimate goal reachability anymore as its understanding of temporal distances is better.

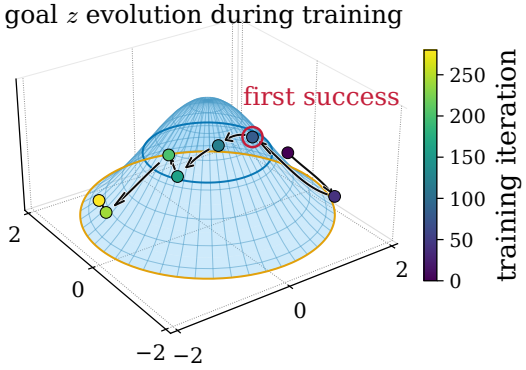


Figure 12: Evolution of $z = f_\phi(g; s_0, a_0)$ during training. It starts far away from origin, at success it is very close to origin indicating over confidence about goal reaching and eventually stabilizing in a high density region.

C.1.1 Action preimage matching baseline results

In this section, as ask the question is a state density estimator the only key for emergent exploration or is the particular choice of policy update using preimage *state* matching as opposed to preimage *action* matching is essential for our method to work. A goal preimage action matching algorithm would update the policy to match the following action distribution:

$$\pi(a \mid s) = \frac{\pi_{\text{old}}(a \mid s)\, p_\gamma^{\pi_{\text{old}}}(g \mid s, a)}{Z} \tag{34}$$

Action distribution matching objectives are much simpler in practice. A behavioral cloning objective such as the following would do action distribution matching:

$$\max_\theta \mathbb{E}_{a \sim \pi_{\text{old}}(a|s)}[p^{\pi_{\text{old}}}(g \mid s, a) \log \pi_\theta(a \mid s)] \tag{35}$$

In order to implement this we use the Maximum Apriori policy objective (Abdolmaleki et al., 2018) that casts RL as an inference problem. we turn it into a goal conditioned algorithm that learns a

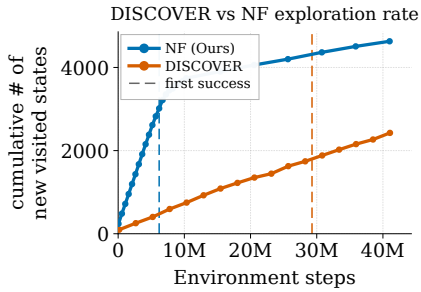


Figure 13: In our algorithm, the exploration rate significantly drops after success, while in DISCOVER it remains nearly unchanged.

variational action distribution that approximates $q(a \mid s) = \frac{\pi_{\text{old}}(a|s)\ \exp(p_\gamma^{\pi_{\text{old}}}(g|s,a))}{Z}$ and use that to do a behavioral cloning of the form $\max_\theta \mathbb{E}_{a\sim q(a|s)}[\log \pi_\theta(a \mid s)]$, so their objective resembles an action preimage matching algorithm. For estimating $p_\gamma^{\pi_{\text{old}}}(g \mid s, a)$ we use contrastive RL. Our results in Fig 14 shows that across all tasks we get much better performance with PIM except for peg insertion where action matching succeeds faster at first but then plateaus at a suboptimal performance.

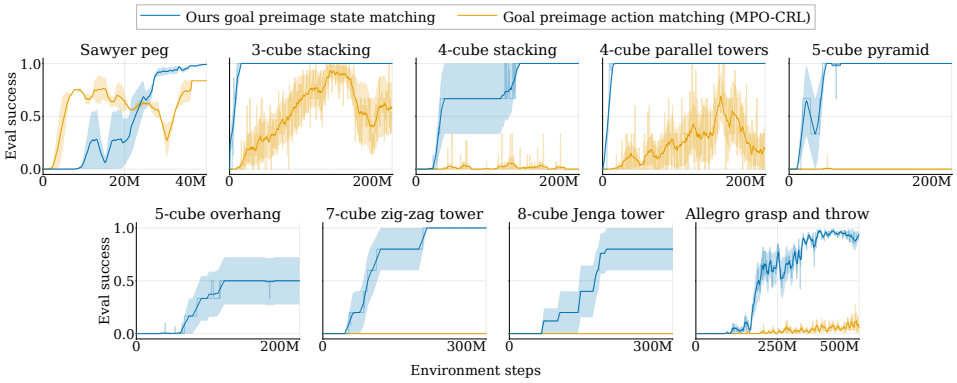


Figure 14: The choice of goal preimage *state* distribution matching as opposed to *action* matching for the policy optimization is the key for our algorithm to work. MPO is an action distribution matching policy improvement algorithm and it underperforms PIM in all tasks except for one.

C.2 Task Details

Builderbench. We use the short-horizon version with a PD controller (Yan, 2026), where the controller determines the forces applied to the cubes based on waypoint actions learned by the policy. Each environment timestep comprises five PD-control timesteps. This short-horizon version of BuilderBench (Ghugare et al., 2026) was designed to make it possible to train a high-level planner that can solve structures with many blocks without having to learn low-level control. The motivation is that prior work has shown that separating control across timescales and levels of abstraction is important for solving complex tasks (Sutton et al., 1999b; Vezhnevets et al., 2017). We use this version of the benchmark to focus on high-level planning in tasks involving many blocks, since low-level object-manipulation skills are already evaluated in our other benchmarks (Sawyer Peg and AllegroHand). The main advantage of this benchmark is that we can arbitrarily increase task difficulty by adding more blocks while still keeping the task feasible for a single trainable policy. The state observation consists of the 3D position of each of the n blocks and one additional dimension indicating the most recently selected block, giving a $(3n + 1)$-dimensional state. In the original BuilderBench formulation (Yan, 2026), the action consists of one scalar cube-selection dimension and three dimensions specifying the waypoint for the selected cube. The selection dimension lies in $[-1, 1]$, which is divided into n equal intervals corresponding to the n cubes. The policy uses a

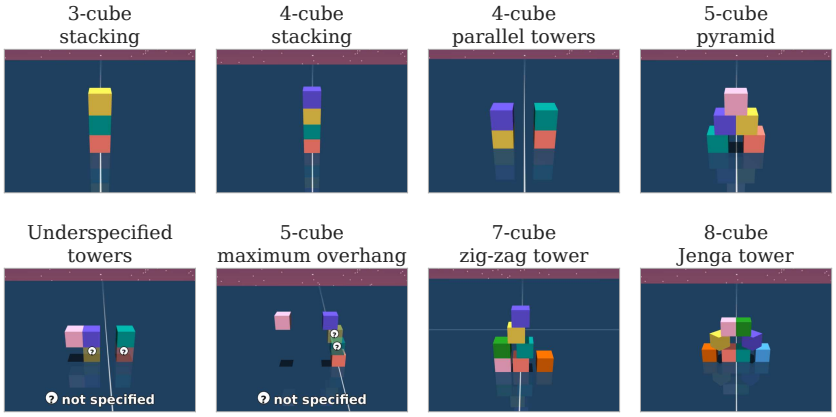


Figure 15: Builderbench tasks we tried. The 5 cube maximum overhang and under-specified towers task only specifies the goal partially, and the agent should figure out the right position for the other blocks to build stable constructions. The 7 cube zig-zag tower requires using a two cube tower as scaffolding for the zig-zag tower.

tanh-Gaussian actor, so the continuous actions are bounded to $[-1, 1]$ and the environment wrapper maps them onto the minimum and maximum allowed value along each direction.

In our experiments, we found it more stable to replace the scalar cube-selection action with an n-way categorical head. The original scalar representation is inherently discontinuous: a small change in the selector near an interval boundary can abruptly switch the selected cube. We therefore use a categorical distribution for cube selection and a tanh-Gaussian distribution for the three continuous waypoint dimensions. We use the same actor architecture for the baselines whenever it helped their performance. The goal is $3 \times n$ dimensional, indicating the goal location of the center of mass of all cubes.

Figure 15 shows all Builderbench tasks we tried. The 5-cube maximum overhang and under-specified towers task only specifies the goal partially, and the agent should figure out the right position for the other blocks to build stable constructions. In both tasks, one cube should be held in the air at the goal position, which forces the agent to find a stable base for the other cubes, because at any moment the agent can only hold one cube. The 7-cube zig-zag tower requires using a two-cube tower as scaffolding for the zig-zag tower. The success is measured permutation-free.

Sawyer peg. This is a Mujoco-based peg insertion task where the agent should put the peg inside a narrow hole inside a box. The position of the peg and object and the gripper at initialization are randomized. The state is 7 dimensional: 3 dims for the peg center location, 3 dim for the gripper center position, and 1 dim for the openness of the drawer. The action is 4 dimensional, specifying the position and openness of the gripper. The goal is 7 dimensional as well, specifying the peg to be inside the hole and gripper standing above it.

ManiSkill. We use a ManiSkill cabinet drawer opening task featuring a Fetch mobile manipulator, where the agent is initialized 1.6 to 1.8 meters away from the cabinet and must drive forward, grasp the handle, and pull the drawer open to at least 75% of its maximum extension. The state space is 10-dimensional, comprising the robot's 3D tool center point, 1D drawer position, 1D gripper aperture, a 1D binary grasping flag, 2D base heading, and 2D mobile base position. The action space is a 9-dimensional continuous box in $[-1.0, 1.0]$, which consists of the delta position control for the arm (3 dims), gripper (1 dim), torso/head (3 dims), and mobile base (2 dims). The goal space is 6-dimensional, specifying the 3D target coordinate of the drawer handle, the 1D target drawer joint position, and fixed targets for releasing the handle with a fully open gripper.

Allegro hand. We use NVIDIA Isaac Gym's AllegroKukaThrow: a 7-DoF Kuka iiwa7 with a 16-DoF Allegro hand (23-DoF actions) that must grasp a cube and throw or place it into a fixed bucket. We use this environment because even relatively simple control objectives become challenging in its high-dimensional state and action spaces, and because of that, prior works heavily rely on carefully

Table 1: PPO + conditional RealNVP hyperparameters goal-preimage matching (PIM).

Hyperparameter	Value
PPO	
Parallel environments	1024
Discount γ	0.99
GAE λ	0.95
PPO epochs / minibatches	10 / 4
Clip ε	0.2
Value coef. c_v	0.5
Entropy coef.	$0.05 \to 0.01$ (linear)
Actor / critic learning rate	3×10^{-4} (linear anneal)
Max gradient norm	0.5
Actor min. std	10^{-5}
Policy / value MLP	6×256
Advantage / return normalization	yes / yes
Observation normalization	no
Rollout length	episode length
Normalizing flow $p_\theta(g \mid s, a)$	
SA encoder	3 layers, width 192
Conditioning dim	64
RealNVP blocks / coupling width	6 / 192
Density steps per PPO iter	10
Batch size	256
Encoder LR (Adam)	3×10^{-4}
Flow LR (AdamW) / weight decay	10^{-4} / 10^{-6}
Flow grad clip	1.0
Goal noise σ / std floor	0.05 / 0.02
Goal whitening $(g - \mu)/\sigma$	yes
Reward EMA τ	0.5
Replay capacity	10^7
Dual ∇_s regularizers	
NF constraint c on $\mathbb{E}[\|\nabla_s \log p\|]$	100
Value constraint c on $\mathbb{E}[\|\nabla_s V\|]$	100
Initial λ / dual LR	5×10^{-4} / 10^{-6}

engineered dense rewards (Handa et al., 2023; Petrenko et al., 2023). Our goal is to test whether the reward obtained directly from our NF-based density estimator can provide a sufficient learning signal without such hand-designed reward shaping. The allegroHand

The observation is joint positions and velocities plus object xyz, with a compact goal (bucket xyz). Success is the cube center inside the bucket cylinder.

C.3 Hyper-parameters

Tables 2 and 1 report the hyperparameters used for MetaWorld and for all other tasks, respectively. MetaWorld is CPU-based, so we cannot scale the number of parallel environments as much as in the other tasks. We therefore use slightly different hyperparameters that work better with the slower rate of data collection. In particular, we use a smaller NF, which we found works better when data arrives more slowly. The slower data rate also removes the need for NF and value-function regularization, since these regularizers were mainly useful for preventing overfitting when the same data was reused more aggressively.

C.4 Density estimator ablation

Contrastive RL (CRL). We use the Monte Carlo InfoNCE objective of Eysenbach et al. (2022). Given a batch of state-action pairs (s_i, a_i) and corresponding future states $s_{f,i} \sim p^\pi_\gamma(\cdot \mid s_i, a_i)$, we define

$$f(s, a, s_f) \coloneqq \phi(s, a)^\top \psi(s_f)$$

Table 2: PPO + conditional RealNVP specific hyperparameter used for metaworld.

Hyperparameter	Value
PPO	
Parallel environments	4
Entropy coef.	0.005
Rollout length	256
Normalizing flow $p_\theta(g \mid s, a)$	
SA encoder	2 layers, width 128
Conditioning dim	32
RealNVP blocks / coupling width	4 / 128
Reward EMA τ	0.85
Dual ∇_s **regularizers**	
NF constraint c on $\mathbb{E}[\|\nabla_s \log p\|]$	unconstrained
Value constraint c on $\mathbb{E}[\|\nabla_s V\|]$	unconstrained

and minimize

$$\mathcal{L}_{\text{CRL}}(\phi) = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{\exp\left(f(s_i, a_i, s_{f,i})\right)}{\sum_{j=1}^{N} \exp\left(f(s_i, a_i, s_{f,j})\right)} + 0.01 \frac{1}{N}\sum_{i=1}^{N} \left[\log \sum_{j=1}^{N} \exp(f(s_i, a_i, s_{f,j}))\right]^2 \tag{36}$$

where $s_{f,i}$ is the positive future state for (s_i, a_i) and $\{s_{f,j}\}_{j\neq i}$ serve as negative samples from the marginal future-state distribution. We use the 6 layer residual connection MLP for both ϕ and ψ networks. We use $\phi(s,a)^\top \psi(g)$ as the log-likelihood reward in PPO. **TD-InfoNCE.** We use the TD-InfoNCE objective of Zheng et al. (2024) which uses TD bootstrap to obtain a TD version of CRL (Eysenbach et al., 2022). Their objective is:

$$\mathcal{L}^{\text{TD-InfoNCE}}(f) \coloneqq \mathbb{E}_{(s,a)\sim p(s,a),\, s_{t+}^{(2:N)}\sim p(s_{t+})} \left[(1-\gamma)\mathbb{E}_{s_{t+}^{(1)}\sim p(s'|s,a)} \left[\log \frac{\exp\left(f(s,a,s_{t+}^{(1)})\right)}{\sum_{i=1}^{N}\exp\left(f(s,a,s_{t+}^{(i)})\right)} \right] \right.$$
$$\left. + \gamma \mathbb{E}_{\substack{s'\sim p(s'|s,a) \\ a'\sim\pi(a'|s') \\ s_{t+}^{(1)}\sim p(s_{t+})}} \left[\left\lfloor w\left(s', a', s_{t+}^{(1:N)}\right) \right\rfloor_{\text{sg}} \log \frac{\exp\left(f(s,a,s_{t+}^{(1)})\right)}{\sum_{i=1}^{N}\exp\left(f(s,a,s_{t+}^{(i)})\right)} \right] \right]$$

where:

$$f_\phi(s,a,s_f) \coloneqq \phi(s,a)^\top \psi(s_f)$$

and

$$w\left(s,a,s_{t+}^{(1:N)}\right) \coloneqq \frac{\exp\left(f(s,a,s_{t+}^{(1)})\right)}{\frac{1}{N}\sum_{i=1}^{N}\exp\left(f(s,a,s_{t+}^{(i)})\right)}$$

and $\lfloor . \rfloor_{\text{sg}}$ is teh stop-gradient sign. We found that using a log-sum-exp regularizer as in Eq. 36 helps, so we added that too. This loss is a contrastive loss; however, it bootstraps estimations of goal reachability for s', a' to update goal reachability for s, a. We use $\phi(s,a)^\top \psi(g)$ as the log-likelihood reward in PPO.

Universal Value Density Estimation (UVD). Goal reaching is equivalent to maximizing the sparse reward

$$r(s) = \mathbb{1}[s = g]$$

For continuous state spaces, prior work (Schroecker & Isbell, 2020; Diaz-Bone et al., 2025; Andrychowicz et al., 2017) has used a smoothed version of this reward by introducing a tolerance ε,

$$r(s,g) = \mathbb{1}\{\|s-g\|_2 \leq \varepsilon\}$$

Using this reward, one can apply TD-style (Schroecker & Isbell, 2020; Fujimoto et al., 2018) value learning to learn a goal-conditioned Q-function. In this case, $Q_\phi(s_f, s, a)$ estimates the discounted

probability of reaching states within an ε-neighborhood of the future state s_f, and in particular provides an estimate of goal-reaching probability when $\|s_f - g\|_2 \leq \varepsilon$. The corresponding TD objective is

$$\mathcal{L}_\phi^{\text{UVD}} := \mathbb{E}\left[\left(Q_\phi(s_f, s, a) - \left(\mathbb{1}\{\|s' - s_f\|_2 \leq \varepsilon\} + \gamma \bar{Q}_{\phi_{\text{old}}}(s_f, s', a')\right)\right)^2\right] \tag{37}$$

where $\bar{\cdot}$ is the stop gradient sign and ϕ_{old}. This gives us a smoothened occupancy measure estimation i.e., $Q(s_f, s, a) = \int_{\|s_f - \hat{s}_f\| \leq \varepsilon} p_\gamma^\pi(\hat{s}_f)\, d\hat{s}_f$ which is a good estimation of $p_\gamma^\pi(s_f)$ if ε is small. Our UVD estimator uses the same 6 layers MLP network with residual connections as described in C.5. We use $\log Q(g, s, a)$ as PPO reward.

C.4.1 Reward heatmaps

In Fig. 16 we compare the log-likelihood reward from different density estimators before any of the methods reach the goal. We are interested in this phase because we want to understand how the reward changes before success and whether it gives the policy useful directions for exploration. At iteration 5, CRL has the best reward shape, with the reward increasing toward the goal, while UVD has the worst one and its reward even decreases toward the goal. After more updates, at iteration 30, NF gives the best reward shape, with a smooth increase toward the goal. UVD is still the worst, since its reward is not smooth and becomes high only very close to the goal. This makes it less useful for making progress before the first success.

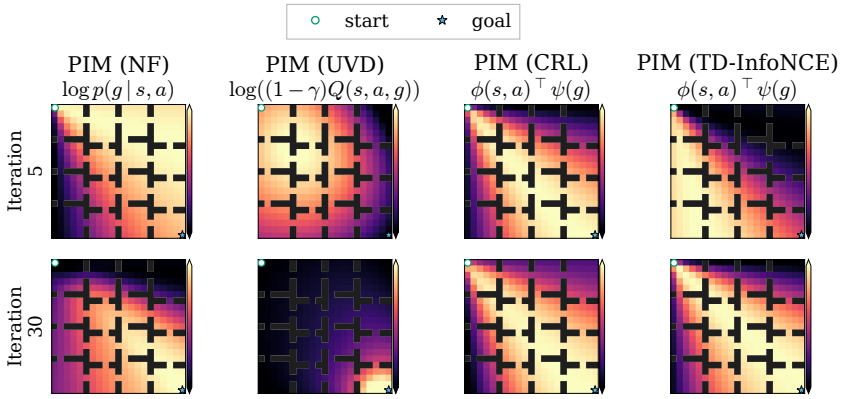


Figure 16: Evolution of the log-likelihood reward for different density estimators before any method reaches the goal. At iteration 5, CRL provides the best-shaped reward, which generally increases toward the goal, while UVD produces the least useful reward. By iteration 30, the NF reward becomes the most well-shaped, with a smooth increase toward the goal. UVD reward stays non-smooth and becomes large only very close to the goal, providing little guidance for progress before success.

C.5 Architecture and training details

All neural networks, except for the NF, which requires an invertible architecture, use a six-layer residual MLP with hidden dimension 256. Each layer consists of a linear transformation followed by LayerNorm and a Swish activation, with skip connections every two layers. We use this architecture because Wang et al. (2025) found it particularly effective for scaling unsupervised RL. We use the same architecture for the policy and value networks in PPO, the representation networks with $d_{\text{repr}} = 64$, the other density estimators, and across all baselines.

Normalizing flow We use the NF architecture introduced in (Ghugare & Eysenbach, 2025), which is inspired by (Dinh et al., 2015; 2017). The NF architecture should be invertible to make sampling $x = f_\phi^{-1}(z)$, $z \sim \mathcal{N}(0, I)$ possible. The architecture consists of multiple consecutive coupling layers. In particular, let $a_\phi^\ell, b_\phi^\ell : \mathbb{R}^{d_{\text{in}}/2} \times \mathbb{R}^{d_{s,a}} \to \mathbb{R}^{d_{\text{out}}/2}$ denote the translation and scale functions of the ℓ-th coupling layer. Let s_f^ℓ denote the input to the ℓ-th block, with (s, a) as the conditioning

inputs. The coupling transformation is given by

$$
\begin{aligned}
s^\ell_{f,1}, s^\ell_{f,2} &= \mathrm{split}(s^\ell_f), \\
\tilde{s}^\ell_{f,2} &= \left(s^\ell_{f,2} + a^\ell_\phi(s^\ell_{f,1}, s, a)\right) \odot \exp\left(b^\ell_\phi(s^\ell_{f,1}, s, a)\right), \\
\tilde{s}^\ell_f &= \mathrm{concat}\left(s^\ell_{f,1}, \tilde{s}^\ell_{f,2}\right),
\end{aligned}
$$

where $\odot$ denotes element-wise multiplication.

This transformation is easily invertible (Ghugare & Eysenbach, 2025), and its log-determinant Jacobian is $\log\left|\det \frac{\partial \tilde{s}^\ell_f}{\partial s^\ell_f}\right| = b^\ell_\phi(s^\ell_{f,1}, s, a)^\top \mathbf{1}$. Each coupling block is preceded by a generalized permutation layer. The permutation layer uses a multiplier matrix

$$W = PL_\phi U_\phi,$$

where P is a fixed permutation matrix, L_ϕ is lower triangular with ones on the diagonal, and U_ϕ is upper triangular. This parameterization is chosen because it allows efficient inversion using triangular matrix operations (Papamakarios et al., 2021). Moreover, the log-determinant has the simple form

$$\log|\det W| = \sum_i \log|(U_\phi)_{ii}|.$$

We additionally use a state-action encoder implemented as a three-layer MLP with hidden dimension 192, output dimension 64, Swish activations, and LayerNorm. We use 6 coupling layers. The translation and scale networks, a_ϕ and b_ϕ, are each three-layer MLPs with hidden dimension 192, LayerNorm in the hidden layers, and LeakyReLU activations. The latent (z) dimension is the same as the input dimension. Overall, this architecture has around $0.76M$ parameters. For all tasks except for metaworld tasks, we use this architecture; for metaworld tasks, we found that a smaller network is more effective; in particular, we use 2×128 state-action encoders with a representation size of 32, 4 coupling blocks, and a hidden dimension of 128 for the a_ϕ and b_ϕ networks. For the NF training psedocode see App. B.6.

Goal-space normalization. Prior work has found input normalization to be helpful for training normalizing flows (Ghugare & Eysenbach, 2025). So we also, we normalize each coordinate using the mean and standard deviation of hindsight goals sampled from the current replay buffer before passing goals to the flow. At each policy iteration, we sample a fresh batch of hindsight goals s_f and compute $\mu = \mathbb{E}[s_f], \quad \sigma = \max\left(\mathrm{Std}(s_f), \sigma_{\min}\right)$ with $\sigma_{\min} = 0.02$. We then use $\tilde{g} = \frac{g-\mu}{\sigma}$ both when training the density model and when computing the PPO reward. The minimum standard deviation prevents coordinates with almost no variation, such as a fixed target height, from being scaled too much. We also include an equal number of copies of the current task goal when computing these statistics, which prevents the queried goal from being too far out of distribution after normalization. The same (μ, σ) are used for density training and reward computation. Adding the task goal (final hard goal) to the normalization batch is very, very important; otherwise, the log-likelihood rewards might explode early in training when g is very out of distribution.

Training noise. During density training, we also add isotropic Gaussian noise to the normalized goals, $\tilde{g} \leftarrow \tilde{g} + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma_n^2 I)$ with $\sigma_n = 0.05$. This follows prior work that uses noise injection to improve generalization and prevent the flow from concentrating too sharply around a finite set of training samples (Ghugare & Eysenbach, 2025). Since the noise is added after normalization, $\sigma_n = 0.05$ corresponds to only a small perturbation relative to the standard deviation of each coordinate. We add this noise only during density training and use the clean normalized task goal when computing the PPO reward, $r(s, a) = \log p_\phi(g \mid s, a)$

C.6 NF Regularization

In order to obtain smooth rewards and avoid the overfitting problem reported in Sec. 5.4 we tried other regularizations, and at the end the state gradient regularizer reported in Sec. 5.4 worked best. Here we discuss other choices we tried that are popular in the literature but weren't as successful as state gradient for us. We hypothesized that TD losses maintain smoother rewards because the

density estimators' parameters are constantly regularized toward the historical moving-averaged version of their own, so we tried regularizing $\|\log p_\phi(s_f \mid s, a) - \log \hat{p}_{\bar{\phi}}(s_f \mid s', a')\|_2$ where $\bar{\phi}$ is the historical time-averaged parameters and $\hat{\cdot}$ is the stop gradient sign; Note that this regularizer does two different smoothening 1) makes the estimates for temporally close states similar 2) doesn't allow the parameters to change too fast. While this regularization helped with stabilization too, we found out that regularizing toward older parameters slowed down success in harder tasks; therefore, in our final results we didn't use this term. So overall insight is that density-based rewards require a smoothening term in the state space but not smoothening in time and smoothening in time can even hurt the performance.

Finally, we noticed that in harder builderbench tasks regularizing $\mathbb{E}\big[\,\|\nabla_s \log p_{\gamma,\phi}(s_f \mid s, a)\|_2\,\big]$ is not enough to stabilize success. Note that this regularizer doesn't directly regularize the reward $\|\nabla_s \log p_{\phi,\gamma}(g \mid s, a)\|$ (evaluated at task goal) and instead regularizes the gradient norm for replay buffer goals s_f. So it does smoothened the reward but in a more indirect way. We realized adding a similar regularizer to the PPO value function i.e., $\mathbb{E}\big[\,\|\nabla_s V(s)\|_2\,\big]$ helps a lot more with directly smoothening the reward. So at the end we used both the log-likelihood regularizer and the value gradient norm regularizer in our experiments but it is worth noting that most tasks get mostly stable success without the value regularizer and adding it was only strictly necessary for 7 and 8 cube builderbench tasks.

C.7 Hyper-parameters

C.8 Baseline Implementation details

In all baselines we use the same 6 layer MLP with layer norm and residual connection architecture used for our own method to keep comparison fair.

RND. Our RND implementation follows (Huang et al., 2022) in its overall structure. PPO hyper-parameters, including the clipping ratio, entropy coefficient, number of environments, and rollout length, are matched to those used by our method (Table 1). Certain other hyperparameters such as epochs and number of minibatches (8/32) and observation normalization (ON) are changed to improve performance.

Single-goal contrastive RL (SGCRL) SGCRL is goal-conditioned contrastive RL with one fixed task goal g for both data collection and evaluation (Liu et al., 2025). The key difference with our method is that their actor update is off-policy, their method only works with CRL as density estimator and that they train policies for all goals not just the task goal. The critic is in-batch InfoNCE on $L_{ij} = \phi(s_i, a_i)^\top \psi(g_j)$ plus a $0.01\,(\log \sum_j e^{L_{ij}})^2$ penalty, with representation size 64. We use 1024 environments, episodes of length 150, 64 critic and actor updates per iteration, and a replay of size 10^7 (learning starts at 10^4).

DISCOVER We observe that in our task, if we only use the goal similarity metric to pick subgoals, and not using the reachability and novelty terms helps with performance. In builderbench tasks we replace the original gaussian tanh policy network of the paper with our policy architecture that uses a categorical actor for selecting which cube to move. We use 1024 environment instances to match the setup with our own implementations.

ICM For the ICM baseline Pathak et al. (2017), we reuse the same PPO implementation as our PPO+RND baseline, replacing only the intrinsic reward with the standard ICM bonus (forward prediction error in the learned feature space, with the usual inverse-model regularizer). The policy is trained on a mixture of sparse task success and this bonus, using the same dual value heads, return normalization, and PPO hyperparameters as RND. We use the original loss mix $\beta = 0.2$ Pathak et al. (2017), 256-dimensional ℓ_2-normalized features, and clip the bonus to $[0, 1]$; on continuous-control tasks the inverse model is trained with mean-squared error rather than a discrete action classifier.

Dirac state matching To match our BuilderBench setup we collect with 1024 environments instead of the paper's single environment. The paper fits the agent kernel density estimator on the whole rollout; at 1024 environments, that pairwise kernel does not fit, so we fit it on 1024 states subsampled from the latest rollout.

C.9 Didactic experiments studying the out-of-distribution generalization of different density estimators

To compare the out-of-distribution behavior of three losses—cross-entropy (CE), contrastive (CL), and temporal-difference (TD)—we use a simple supervised learning over-parameterized linear model. Ce represents out maximum livelihood NF model, CL represent out contrastive RL method and TD represents our UVD objective for density estimation. The point of this didactic experiment is the same question we care about for preimage estimation: when two density estimators share a parameterization but use different losses, do they learn different functions, and do those functions generalize differently to unseen goals g?

Experiment setup There are $N = 100$ training points in dimension $d = 200$, so $N \ll d$. Each point x_i has a target distribution p_i over 5 labels $y_i \in \{1, 2, 3, 4, 5\}$, with $\sum_{j=1}^{5} p_{ij} = 1$.

We first sample a random immediate distribution q_i over the same 5 labels. The targets p_i are then defined by the TD recursion

$$p_{ij} = (1 - \gamma)\, q_{ij} + \gamma\, p_{i+1,j}, \qquad p_N = q_N. \tag{38}$$

This makes p_i a valid target for a TD loss as well as for supervised losses.

The model is $\hat{p}_i = \mathrm{softmax}(x_i W)$. Because $N \ll d$, many matrices W can fit the training data. We train the same model with each loss and compare the solutions $W^\star$: whether they differ, and how they behave on out-of-distribution x. The training objective for each loss is as follows:

Let $\hat{p}_i = \mathrm{softmax}(x_i W)$.

$$\mathrm{CE}: \quad \min_W \mathbb{E}_i \left[-\sum_j p_{ij} \log \hat{p}_{ij} \right]. \tag{39}$$

$$\mathrm{TD}: \quad \min_W \mathbb{E}_{i<N} \left[\left\| \hat{p}_i - \big((1-\gamma)\, q_i + \gamma\, \mathrm{sg}(\hat{p}_{i+1}) \big) \right\|_2^2 \right] + \left\| \hat{p}_N - q_N \right\|_2^2. \tag{40}$$

$$\mathrm{CL\ (InfoNCE)}: \quad \min_W \mathbb{E}_i \left[-\sum_j p_{ij} \log \frac{\exp((x_i W)_j)}{\exp((x_i W)_j) + (B-1)\, p_j^{\mathrm{marg}} \sum_l \exp((x_i W)_l)} \right], \tag{41}$$

where $p_j^{\mathrm{marg}} = \mathbb{E}_i[p_{ij}]$ and $\mathrm{sg}(\cdot)$ is a stop-gradient. We update the parameter W for each of these losses separately using gradient descent until convergence.

Results We observe that all three objective converge exactly to the same W (Figure 17) although TD converges slower. This shows that under stationary data and same parametrization they all have similar out of distribution properties as well since they converge exactly to the same solution.

Takeaway This experiment shows that, the CE, CL, and TD objectives do not necessarily converge to different solutions. CE and CL converge to essentially the same solution, and TD also approaches the same solution asymptotically, although its trajectory during training is different. For CL, the training trajectory also depends on batch size: with large batches, CL closely matches CE throughout training, while with smaller batches the two can differ before converging.

This suggests that the choice of objective alone does not necessarily determine different out-of-distribution behavior. Under the same parameterization, these objectives can converge to the same solution. The better performance of NF in our experiments could therefore come from two other sources. First, NF uses a different architecture, namely an invertible model with coupling layers. Second, even when two objectives have the same asymptotic solution on a fixed dataset, they can follow different learning trajectories. In the online RL setting, this matters because different intermediate density estimates lead to different data collection, so the models are no longer trained on the same data and may eventually converge to different solutions.

The fact that the TD objective converges much more slowly than the two Monte Carlo objectives is also consistent with our empirical results: TD-based density estimators generally learn more slowly and fail on some tasks, such as the 7-cube zig-zag tower, that both Monte Carlo methods can solve.

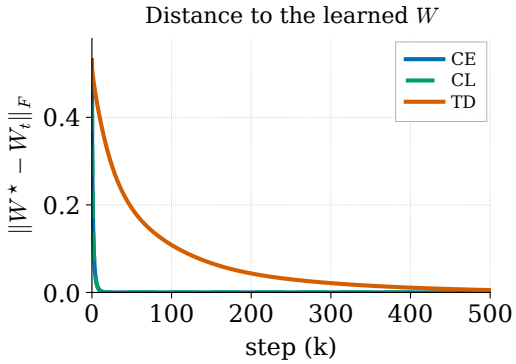


Figure 17: Cross entropy and contrastive and TD loss all converge to the same solution in an over-parameterized supervised learning setup. Therefore all three have similar out of distribution proprieties.

C.10 Comparison with goal Dirac state distribution matching

State matching for goal reaching has been explored in prior works (Durugkar et al., 2021; Agarwal et al., 2023; Lee et al., 2019; Sikchi et al., 2025). However, the fundamental difference with our work is that, instead of using a dynamically evolving goal-preimage distribution, these methods consider a fixed target distribution, $\delta_g(s_f)$ (a Dirac delta at the goal), which places all of its mass on the goal state, and attempt to match this distribution from the beginning of training, either through f-divergence matching or Wasserstein matching. Our empirical results in Fig. 7, 10 shows that these algorithms cannot solve long-horizon exploration tasks. Here we do a conceptual analysis of these methods using didactic experiments. Our analysis show that (i) these methods, before reaching the goal, act as a novelty-seeking method as opposed to our method that dynamically forms hypotheses about reaching the goal and executes them (ii) the objective functions of these methods are extremely non-smooth: they are almost flat before the goal is discovered and exhibit a sudden jump upon first reaching it, making them unsuitable for gradient-based optimization.

$$\min_{\pi} D_{\mathrm{KL}}\left(p^{\pi}_{\gamma}(s) \| q_{\varepsilon}(s)\right), \qquad q_{\varepsilon}(s) \propto \begin{cases} 1 & s = g, \\ \varepsilon & s \neq g. \end{cases}$$

Here, $\varepsilon > 0$ assigns nonzero mass to all non-goal states, making the KL objective finite and computable in practice. This objective is equivalent to

$$\begin{aligned} D_{\mathrm{KL}}\left(p^{\pi}_{\gamma}(s) \| q_{\varepsilon}(s)\right) &= \sum_{s \neq g} p^{\pi}_{\gamma}(s) \log \frac{p^{\pi}_{\gamma}(s)}{\varepsilon} + p^{\pi}_{\gamma}(g) \log p^{\pi}_{\gamma}(g) \\ &= \sum_{s \neq g} p^{\pi}_{\gamma}(s) \log \frac{1}{\varepsilon} + \sum_{s} p^{\pi}_{\gamma}(s) \log p^{\pi}_{\gamma}(s) \\ &= -\log \frac{1}{\varepsilon} p^{\pi}_{\gamma}(g) - H(S) + c, \end{aligned}$$

where c is independent of π and $H(S)$ is the state distribution entropy. where c is a constant indep of π. Therefore, the policy optimization problem reduces to a combination of state-entropy maximization and goal reaching. There are two major issues with this objective. First, its exploration behavior is driven entirely by the state-entropy term, and therefore encourages exploration through seeking novel states, similarly to other novelty-based methods. This is evident by considering the results of Agarwal et al. (2023) which show that their objective is equivalent up to first-order derivatives with a reward maximization objective with reward $r(s) = \log \frac{q_{\varepsilon}(s)}{p^{\pi}_{\gamma}(s)}$. Before seeing the goal, the numerator is always fixed at ε, and the reward turns into a pure novelty-seeking reward. In Section 5, we show that this type of exploration is insufficient for solving challenging long-horizon tasks and is only effective on simpler tasks. Second, this objective does not guarantee convergence to the optimal policy unless ε is very small. Figure 18 (Left) illustrates this in a didactic example, showing how increasing ε causes the converged policy to deviate further from optimality. It may

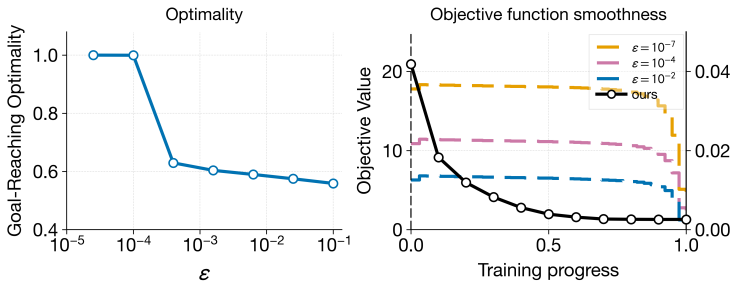


Figure 18: Left figure: Distribution matching against the target Dirac function at the goal, only recovers the optimal policy for very small choices of ε (an algorithm hyperparameter), but at the same time choosing small ε results in an extremely non-smooth objective function that doesn't provide any learning signal until very late in training (Right figure). Our algorithm instead avoids such hyperparameter and, by matching a dynamically evolving target distribution, obtains a smooth objective with a useful learning signal throughout training.

therefore seem beneficial to choose a very small ε. However, Figure 18 (Right) shows that as ε decreases, the objective becomes increasingly non-smooth during the evolution from the initial policy to the optimal policy. This plot shows that the objective provides little gradual feedback to policy optimization before the policy is already very close to success. In contrast, since the goal-preimage distribution evolves with the policy, the objective (Equation 18) changes smoothly during training and provides gradual updates (Figure 18 - Right).