SKILL-BASED REINFORCEMENT LEARNING WITH INTRINSIC REWARD MATCHING

Anonymous authors
Paper under double-blind review

ABSTRACT

While unsupervised skill discovery has shown promise in autonomously acquiring behavioral primitives, there is still a large methodological disconnect between task-agnostic skill pretraining and downstream, task-aware finetuning. We present Intrinsic Reward Matching (IRM), which unifies these two phases of learning via the *skill discriminator*, a pretraining model component often discarded during finetuning. Conventional approaches finetune pretrained agents directly at the policy level, often relying on expensive environment rollouts to empirically determine the optimal skill. However, often the most concise yet complete description of a task is the reward function itself, and skill learning methods learn an *intrinsic* reward function via the discriminator that corresponds to the skill policy. We propose to leverage the skill discriminator to *match* the intrinsic and downstream task rewards and determine the optimal skill for an unseen task without environment samples, consequently finetuning with greater sample-efficiency. Furthermore, we generalize IRM to sequence skills and solve more complex, long-horizon tasks. We demonstrate that IRM is competitive with previous skill selection methods on the Unsupervised Reinforcement Learning Benchmark and enables us to utilize pretrained skills far more effectively on challenging tabletop manipulation tasks.

1 INTRODUCTION

Generalist agents must possess the ability to execute a diverse set of behaviors and flexibly adapt them to complete novel tasks. Although deep reinforcement learning has proven to be a potent tool for solving complex control and reasoning tasks such as in-hand manipulation (OpenAI et al., 2019) and the game of Go (Silver et al., 2016), specialist deep RL agents learn each new task from scratch, possibly collecting new data and learning to a new objective with no prior knowledge. This presents a massive roadblock in the way of integration of RL in many real-time applications such as robotic control where collecting data and resetting robot experiments is prohibitively costly (Kalashnikov et al., 2018).

Recent progress in scaling multitask reinforcement learning (Reed et al., 2022; Kalashnikov et al., 2021) has revealed the potential of multitask agents to encode vast skill repertoires, rivaling the performance of specialist agents and even generalizing to out-of-distribution tasks. Moreover, skill-based unsupervised RL (Laskin et al., 2022; Liu & Abbeel, 2021; Sharma et al., 2020) shows promise of acquiring similarly useful behaviors but without the expensive per-task supervision required for conventional multitask RL. Recent skill-based RL results suggest that unsupervised RL can distill diverse behaviors into distinguishable skill policies; however, such approaches lack a principled framework for connecting unsupervised pretraining and downstream finetuning. The current state-of-the-art leverages inefficient skill search methods at the policy level such as performing a sampling-based optimization or sweeping a coarse discretization of the skill space (Laskin et al., 2021). However, such methods still exhibit key limitations, namely they (1) rely on expensive environment trials to evaluate which skill is optimal and (2) are likely to select suboptimal behaviors as the continuous skill space grows due to the curse of dimensionality.

In this work, we present Intrinsic Reward Matching (IRM), a scalable algorithmic methodology for unifying unsupervised skill pretraining and downstream task finetuning by leveraging the learned intrinsic reward function paramterized by the skill discriminator. Centrally, we introduce a novel approach to leveraging the intrinsic reward model as a multitask reward function that, via

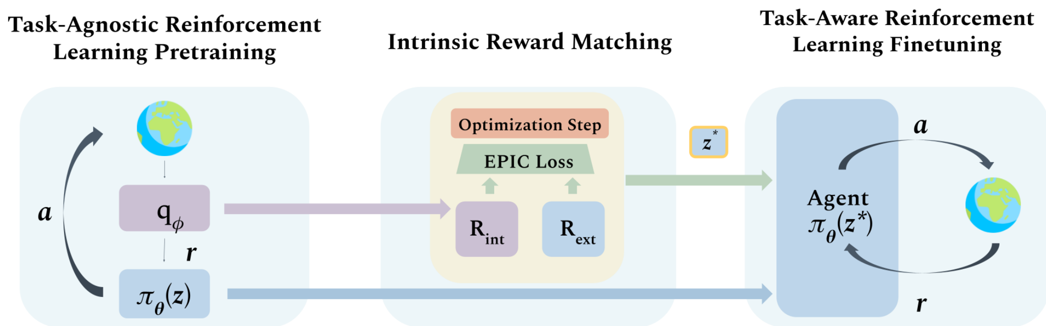


Figure 1: Intrinsic Reward Matching (IRM) Framework. IRM takes place in three stages: (1) Task-agnostic RL pretraining learns skill primitives in conjunction with a skill discriminator. (2) With no environment interaction, IRM minimizes the EPIC Loss between the intrinsic reward parameterized by the discriminator and the extrinsic reward with respect to the skill vector z. (3) The skill policy conditioned on the optimal z^* finetunes to task reward to solve the downstream task.

interaction-free task inference, enables us to select the most optimal pretrained policy for the extrinsic task reward. During pretraining, unsupervised skill discovery methods learn a discriminator-parameterized, family of reward functions that correspond to a family of policies, or skills, through a shared latent code. Instead of discarding the discriminator during finetuning as is done in prior work, we observe that the discriminator is an effective task specifier for its corresponding policy that can be *matched* with the extrinsic reward, allowing us to perform skill selection while bypassing brute force environment trials. Our approach views the extrinsic reward as a distribution with measurable proximity to a pretrained multitask reward distribution and formulates an optimization with respect to skills over an optimal-policy invariant psuedometric called EPIC Gleave et al. (2020).

Contributions The key contributions of this paper are summarized as follows:

- We describe a unifying discriminator reward matching framework and introduce a practical algorithm for selecting skills without relying on environment samples (Section 3).
- We demonstrate that our method is competitive with previous finetuning approaches on the Unsupervised Reinforcement Learning Benchmark (URLB), a suite of 12 continuous control tasks (Section 4.1).
- We evaluate our approach on more challenging tabletop manipulation environments which underscore the limitations of previous approaches and show that our method finetunes more efficiently (Section 4.2).
- We generalize our method to sequence pretrained skills and solve long-horizon manipulation tasks (Section 4.3) as well as ablate key algorithmic components.
- We provide analysis and visualizations that yield insight into how skills are selected and further justify the generality of our method (Section 5).

2 Background

2.1 Pretrained Multitask Reward Functions

In describing the IRM framework, we begin with the key observation that the intrinsic reward function learned during skill pretraining can be viewed as a multitask reward function, where the continuous skill code z determines the task. In other words, we have some function:

$$\mathcal{R}^{int}(\tau, z) := \text{VLB}(\tau, z) \tag{1}$$

where VLB $\leq I(\tau, z)$, the variational lower bound proposed in (Barber & Agakov, 2003) (τ is a trajectory representation such as (s, s')). Since skill discovery algorithms aim to maximize $I(\tau, z)$, we can view its parameterized lower bound VLB as a multitask reward function: scoring transitions based on their alignment with a skill code. This interpretation of the intrinsic reward is generally appropriate in skill-based unsupervised RL algorithms, most of which rely on an approximate maximization of a variational lower bound to the mutual information. Notably, this reward function is not learned through any external supervision by hand-designed objectives as in (Reed et al., 2022), but rather arises from an unsupervised RL pretraining phase. For a detailed explanation of the mutual information decompositions of various skill discovery algorithms, refer to Appendix A.2.

2.2 Equivalent-Policy Invariant Comparison

We can formalize a general notion of reward function similarity by equivalent-policy invariant comparison (EPIC) as established in Gleave et al. (2020). EPIC defines a pseudometric between two reward functions that is invariant on an equivalence class of reward functions that always induce the same optimal policy:

$$D_{\text{EPIC}}(R_A, R_B) = D_\rho(C_{D_S,D_A}(R_A)(S, A, S'), C_{D_S,D_A}(R_B)(S, A, S')). \quad (2)$$

where $D_\rho(X,Y) = \sqrt{\frac{1-\rho(X,Y)}{2}}$ is the Pearson distance between two random variables X and Y. We compute the Pearson distance over samples from distributions $P_S \in \Delta(S)$ and $P_A \in \Delta(A)$ where our random variable inputs are rewards computed over these samples. C_{D_S,D_A} is the canonically shaped reward defined as:

$$C_{D_S,D_A}(R)(s, a, s') = R(s, a, s') + \mathbb{E}[\gamma R(s', A, S') - R(s, A, S') - \gamma R(S, A, S')] \quad (3)$$

where $R : S \times A \times S \to \mathbb{R}$ is a reward function and $D_S \in \Delta(S)$ and $D_A \in \Delta(A)$ are distributions over states and actions. S and S' are random variables independently sampled from D_S and A sampled from D_A. The canonicalization ensures *invariance to reward shaping* such that rewards with different shaping but similar semantics are represented similarly. In practice, the final term can be omitted as the Pearson correlation is *invariant to constant shifts and scaling*.

3 Intrinsic Reward Matching

3.1 Task Inference via Intrinsic Reward Matching

A multitask reward function that can supervise the learning of diverse behaviors is useful in its own right. However, in the case of skill-based RL, we have additionally learned a corresponding $\pi(a|s, z)$. Therefore, for any "task" that can be specified our intrinsic reward function, we already have an optimal policy, so long as we condition on the corresponding skill. If we have learned a sufficiently diverse library of skills, we might expect that some of our skills share behavioral similarity to the optimal policy for the downstream task. As is such, it also holds that the corresponding intrinsic reward for that skill is a semantically similar task specification to the downstream task.

Given this interpretation of intrinsic reward, we posit that the task of identifying which our pretrained skills to apply to a downstream task can be reframed as inferring which task in our multitask reward function is most similar to the downstream task. Moreover, we should hope to find the skill code z that produces the reward function most semantically aligned with the downstream task reward.

With this formalism, we can formulate the task inference problem as performing the following optimization:

$$z^* = \arg\min_z D_{\text{EPIC}}(R^{int}(\tau, z), R^{ext}(\tau)) \quad (4)$$

in order to find z^* most aligned with the task reward. Moreover, 4 performs a minimization of a novel loss we name the ***EPIC Loss*** with respect to the skill parameter z. By EPIC's equivalence class invariance, we know that if the EPIC loss is small for some z^*, and $\pi(a|s, z^*)$ is near optimal for $R^{int}(\tau, z^*)$, then $\pi(a|s, z^*)$ approaches the optimal policy for the task as specified by R^{ext}. Notably, we require access to the task reward function R_{ext} to compute the EPIC loss.

Computing R^{int} during reward matching During pretraining, for some methods such as Laskin et al. (2022); Sharma et al. (2020), we require negative samples in order to compute the variational objective 1 and avoid a degenerate optimization where all embedded trajectories have high similarity with all skills. However, during selection when skills are fixed, the negative sampling component amounts to a reward offset which does not impact the task semantics. Furthermore, since we may not in general have access to a large amount of negative samples on a given downstream task, we choose to simplify the objective to the following:

$$\mathcal{R}^{int}(\tau, z) := \text{VLB}(\tau, z) \equiv q_\phi(\tau, z) \quad (5)$$

where q_ϕ is the skill discriminator. This parameterization of the intrinsic reward preserves the alignment semantics of VLB without the normalization by negative samples. For more details regarding the discriminator paramterization of the intrinsic reward for (Laskin et al., 2022; Sharma et al., 2020) refer to Appendix A.3 and Appendix A.4.

Algorithm 1: Intrinsic Reward Matching (IRM)

Require: Downstream task $\mathcal{T}$, D_S, D_A
Require: Pretrained policy $\pi_\theta(a|s,z)$, intrinsic reward $r_{int}(s,s',z)$, and extrinsic reward $r_{ext}(s,s')$ for $\mathcal{T}$.
Require: Optimization $N_{OP} = 5000$ steps and fine-tune $N_{FT} = 100K$ steps.
`/* Skill Selection of` z^* `via EPIC Loss */`
for N_{OP} *steps* **do**
  Sample Pearson samples $S_P, A_P, S'_P \sim P_S, P_A, P_S$.
  Sample Canonical samples $S_C, A_C, S'_C \sim D_S, D_A, D_S$.
  Calculate $D_{\text{EPIC}}(r_{int}(s,s',z), r_{ext}(s,s'))$ with Equation 2.
  Take optimization step (gradient descent, CEM step, etc.) with Equation 4 .
end for
Evaluate zero-shot and finetuning performance of RL agent with z^* on downstream task $\mathcal{T}$

3.2 EPIC Sample-Based Approximation

We make a number of sample-based approximations of various unknown quantities in order to concretize the continuous optimization 4 as a tractable loss minimization problem.

Canonical State and Action Distribution Approximation: In order to canonicalize our reward functions, we estimate the expectation over the state and action distributions $\mathcal{D}_S$ and $\mathcal{D}_A$ respectiveley with a sample-based average over 1024 samples. These distributions can be entirely arbitrary, though using heavily out-of-distribution samples with respect to pretraining can weaken the accuracy of the approximation. We choose to instantiate a uniform distribution bounded by known workspace constraints for both of these distributions. We ablate various choices for D_S amd D_A in Table 6.

Sampling Distribution for Pearson Correlation: In order to compute the Pearson correlation between the canonicalized reward functions in comparison, we require samples from both reward distributions under some arbitrary state and action distributions P_S and P_A respectively. We can use the visitation distribution induced by the policy for this distribution, allowing us to compute the Pearson distance over the batch to estimate the EPIC loss. We can backpropagate through the EPIC loss in order to optimize z with stochastic gradient descent (SGD) or rely on 0th order population based optimization algorithms like Cross-Entropy Method (CEM).

Alternatively, we find that generating samples uniformly within roughly in-distribution workspace bounds as with the reward canonicalization often leads improved approximations. For this reason in addition to *entirely circumventing the need for environment interactions*, we elect to use the latter approach. Furthermore, as both sample generation and relatively inexpensive function evaluation are independent of the online-finetuning phase, we can perform the full skill optimization as a self-contained preprocess to downstream policy adaptation. We ablate various choices for P_S amd P_A in 6. We present the full algorithm in detail in Algorithm 1.

3.3 Generalization to Skill Sequencing

Many realistic downstream tasks derive additional complexity from temporally extended planning horizons. In contrast to hierarchical reinforcement learning (HRL) approaches, which aim to stitch together pretrained skills at the policy level with a higher-level manager policy, we can extend the task matching framework of IRM to efficiently solve the problem of skill sequencing, entirely doing away with the manager policy. Consider the long-horizon setting where we have a sequence of reward functions to optimize over some task horizon H. Central to the finetuning problem is determining over what time intervals should potentially different pretrained skills be selected. In this work we predetermine a fixed skill horizon $\lfloor H/N \rfloor$ where N is the number of rewards. This skill horizon could in principle be specified as a parameter and learned from the task reward signal.

Next, in order to perform skill selection over each time interval, we perform the IRM algorithm in parallel for each reward. We observe that due to the serial dependency of skills executed in sequence, that using on-policy samples or constraining uniform sampling distributions to a tight radius around the resulting state configurations of executing the previous skill at-times yields better skill selections. We should expect this difference when the downstream task is not spatially invariant such as goal reaching, whereas for tasks like directional running either approach should perform similarly. After selecting the skills, we freeze our selections and finetune the skill policies jointly.

4 Experiments

In this section we aim to experimentally evaluate whether IRM improves the adaptation sample-efficiency of skill finetuning on a downstream reinforcement learning task as compared to baselines. For pretraining skills, we experiment with both the CIC (Laskin et al., 2022) and DADS (Sharma et al., 2020) algorithms. We consider *IRM Random* a version of IRM that randomly samples skills and picks the one with the lowest EPIC loss, *IRM CEM* which selects elites as those skills with the lowest EPIC loss, and *IRM Gradient Descent* which minimizes the EPIC loss using the Adam optimizer and uses backpropagation through the discriminator to regress the optimal skill. Skill pretraining hyperparameters are listed in A.6. Skill finetuning hyperparameters are listed in A.7. We do not tune *any* optimization hyperparameters for the IRM methods - doing so would likely further increase performance. We show zero-shot results for skill policies selected as well as finetuning results of selected skills on the downstream task.

Environments We evaluate IRM on URLB (Laskin et al., 2021), which consists of twelve downstream tasks in three challenging continuous control domains in the DMControl suite: Walker, Quadruped, and Jaco with further details in A.5. We also design a reaching and a tabletop pushing environment in the OpenAI Gym Fetch environment (Brockman et al., 2016) with further explanation in 4.2 and implementation details in A.5.

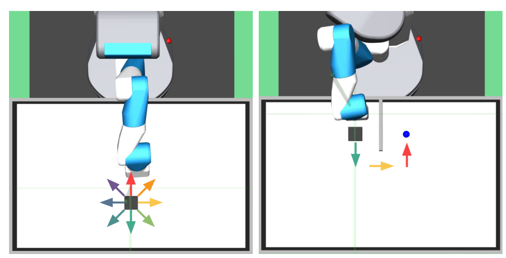


Figure 2: In our Fetch Push environment, we learn skills that roughly correspond to moving the block in different directions. Downstream tasks may involve simple goals or more distant goals that require composition of multiple skills across an extended time horizon and around obstacles.

Baselines We benchmark many conventional finetuning approaches after a single skill pretraining phase of Contrastive Intrinsic Control (CIC) (Laskin et al., 2022). The *Grid Search (GS)* baseline coarsely sweeps each of 10 skills evenly from the all 0's skill vector to the all 1's skill vector and finetunes the skill which achieves the best evaluation reward over an episode. *Env Rollout* randomly samples 10 skills to evaluate with a rollout and *Env Rollout CEM* uses the episode reward as the metric by which to select elites. *Random Skill* selects a skill at random. All baselines use the TD3 (Fujimoto et al., 2018) RL algorithm to eliminate confounding factors when comparing algorithms.

Evaluation We follow an identical evaluation to the 2M pre-training setup in URLB. First, we pretrain each RL agent with the intrinsic rewards for 2M steps. Then, we finetune each agent to the downstream task with extrinsic rewards in the data-efficient regime of 100k steps. Since our primary contribution involves skill selection, we especially focus on zero-shot episode rewards: rewards achieved by a selected skill policy but without any RL updates on the task reward. We report results averaged over 5 seeds and 1 unit of standard error to capture variance for all results.

4.1 Unsupervised Reinforcement Learning Benchmark

In Table 1, we display the zero-shot performance of IRM-based methods compared to interaction-based methods over all 12 URLB tasks. On most of the Walker and Quadruped tasks IRM is either comparable to our outperforms the interaction baselines. Unsurprisingly, methods like IRM GD and IRM CEM tend to perform better than IRM Random which does not have the luxury of iterative refinement of a relatively smooth EPIC loss manifold as shown in Figure 5. We find that neither

Task	IRM CEM	IRM GD	IRM Rand	Env Roll.	Env CEM	GS	Rand
Jaco Top Left	0 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.186 ± 0.11	0.770 ± 0.28	**1.84 ± 0.00**	0.00 ± 0.00
Jaco Top Right	0.086 ± 0.04	0.64 ± 0.24	0.120 ± 0.097	7.34 ± 3.39	9.82 ± 5.25	**16.1 ± 0.00**	3.50 ± 2.50
Jaco Bot. Left	0.052 ± 0.03	0.00 ± 0.00	0.00 ± 0.00	0.175 ± 0.16	**0.408 ± 0.22**	0.102 ± 0.00	0.00 ± 0.00
Jaco Bot. Right	2.48 ± 2.22	0.00 ± 0.00	0.360 ± 0.305	0.086 ± 0.073	**9.07 ± 3.33**	0.191 ± 0.00	0.001 ± 0.001
Walker Stand	**19.9 ± 9.3**	9.75 ± 1.4	12.5 ± 3.0	18.9 ± 3.7	**22.4 ± 4.3**	13.9 ± 4.41	**20.8 ± 7.6**
Walker Walk	5.86 ± 0.34	7.48 ± 0.55	15.5 ± 5.5	14.9 ± 2.912	13.33 ± 3.20	9.40 ± 2.8	**15.6 ± 4.9**
Walker Run	6.82 ± 0.66	7.17 ± 0.28	8.101 ± 0.97	7.92 ± 0.69	5.87 ± 1.2	6.56 ± 1.2	**8.81 ± 1.2**
Walker Flip	20.6 ± 1.2	14.8 ± 1.1	17.3 ± 2.3	**23.8 ± 1.9**	17.3 ± 2.8	21.8 ± 0.00	14.4 ± 1.8
Quadr. Stand	**51.5 ± 11.8**	40.3 ± 11.4	40.15 ± 13.4	40.6 ± 9.7	47.5 ± 8.7	37.4 ± 12.1	44.6 ± 12.6
Quadr. Run	23.9 ± 5.5	**24.4 ± 4.8**	20.2 ± 6.5	20.6 ± 4.4	24.2 ± 4.1	17.3 ± 5.5	21.7 ± 6.2
Quadr. Jump	38.1 ± 8.5	**41.1 ± 9.4**	35.9 ± 10.6	30.5 ± 7.0	36.8 ± 6.3	29.7 ± 9.8	33.3 ± 9.5
Quadr. Walk	17.5 ± 6.2	11.5 ± 3.7	17.07 ± 6.2	19.3 ± 2.7	**25.5 ± 4.0**	9.21 ± 2.0	16.4 ± 5.8
Fetch Reach	95.9 ± 1.0	87.5 ± 0.20	92.5 ± 1.085	85.0 ± 6.2	87.8 ± 1.9	**97.3 ± 0.0**	16.67 ± 19.1
Fetch Push	**80.2 ± 2.5**	73.1 ± 0.48	77.6 ± 2.7	74.3 ± 0.92	75.4 ± 2.6	72.1 ± 0.00	51.5 ± 12.5

Table 1: IRM with various optimization methods compared to environemnt rollout-based skill selection and random skill selection. Despite CIC only able to learn a limited skill library on URLB domains, on many tasks, IRM based methods manage to rival or exceed skill selection baselines that are reliant on expensive environment trials.

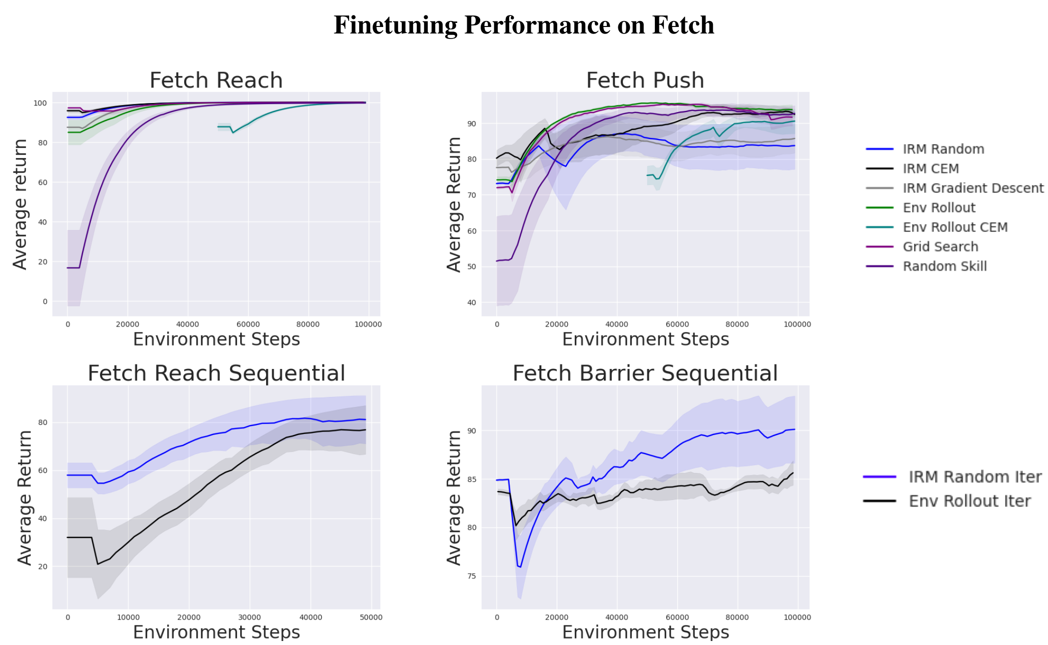


Figure 3: The performance gap between the IRM skill selection methods and random skill selection evidences the sample efficiency gains of bootstrapping a pretrained policy with task-level semantics close to the task reward. IRM-based methods select optimal skills with no environment interaction and consequently finetune efficiently. **Top:** Fetch Reach and Block Push tasks. **Bottom:** Long-horizon Fetch Reach and Block Push with obstacles tasks.

our method nor the baselines are well-suited for skill selection on the Jaco tasks. This is likely because these tasks are very sparsely rewarded, making it unlikely that many samples, either randomly generated as in IRM or rolled out, will consistently result in high rewards.

We note that CIC as well as the other skill pretraining algorithms we experimented with do not yield qualitatively useful skills for some of the downstream tasks evaluated. As a result, we also evaluate on robotic manipulation tasks in the following section where pretraining algorithms learn more interesting behaviors. We also display the finetuning efficiency that results from various approaches for selecting skills in Appendix A.9.

4.2 Tabletop Manipulation

Reach Target We evaluate IRM on the Reach Target task, where the Fetch robot is rewarded for reaching a target position. IRM outperforms or closely matches environment-rollout methods while requiring no environment samples to perform skill selection. As shown in Table 1, the random skill policy performs particularly poorly and with very high variance relative to the IRM and environment-rollout based methods. Moreover, appropriate skill selection is required for strong zero-shot performance as certain skills obtain much higher rewards than others. Figure 3 shows the finetuning performance of the methods on the downstream task reward. IRM-based methods are much more sample efficient in reaching the optimal performance than environment-rollout-based methods. Without environment interactions, IRM is still able to find better skill selections that those methods that require samples, and as a result produces more efficient downstream task adaptation.

Push Block to Goal Next, we evaluate IRM on a more complex manipulation task involving pushing a block to a goal position. We report the zero-shot IRM skill selection performance in Table 1 and finetuning performance in Figure 3. This more complex task similarly benefits from bootstrapping the appropriate pretrained skill policy as evidenced by the performance gap of the selection based methods over random skill selection. We remark that even for more complex manipulation tasks, IRM is robust in consistently guiding optimal skill selection without requiring any interaction with the environment. Although Env Rollout CEM is one of the stronger baselines in terms of zero-shot reward, it exceeds the computational budget of 100k interactions entirely on skill selection. For illustrative purposes, we show the plot starting at 50k to underscore the limitations of relying on environment rollouts for skill selection. This issue is heavily exacerbated for all environment rollout methods as episode horizons increase Vinyals et al. (2019).

4.3 Skill Sequencing for Long-Horizon Tasks

Long-Horizon Manipulation Building on the results in Section 4.2, we demonstrate that IRM fully generalizes to solving long-horizon tasks in the setting of tabletop manipulation. During the unsupervised pretraining phase, skill discovery methods can acquire useful skills such as directional block pushing or pushing the block to certain spatial locations. We show that IRM can intelligently select a sequence of such skills to finetune via reward matching, avoiding learning a hierarchical meta-controller that finetunes at the policy level.

Task	IRM Seq	Env Seq
Fetch Reach Seq	**57.9** ± 5.2	32.0 ± 16.7
Fetch Push Seq	**84.9** ± 0.12	83.687 ± 0.30

Table 2: Zero-shot rewards on long-horizon manipulation tasks

For the Fetch Reach environment, we consider an extended horizon where the agent is tasked with reaching a sequence of goals in a particular order. For the Fetch Push task, we consider the environment depicted in Figure 2, where the agent must navigate around a barrier introduced during the finetuning phase in order to reach the goal. For additional details regarding environment design, refer to Appendix A.5. We compare an extended versions of the IRM and Environment Rollout methods, denoted IRM Seq and Env Seq, described in depth in Appendix A.10. Both methods iteratively select skills. Due to the sequential nature of the task, IRM uses limited environment samples once each skill is selected. In both settings, IRM outperforms the environment rollout method in identifying (Table 2 and finetuning skills (Figure 4.3 Row 2).

Matching Metric Ablations We validate the importance of employing the EPIC pseudometric for formulating the matching loss by ablating its contribution against more naive selections in Table 4. L1 and L2 losses are common metrics in supervised regression problems but are poor choices for comparing task similarity with rewards. Moreover, rewards can have arbitrary differences in scaling and shaping that L1 and L2 are not invariant to. To strengthen these comparisons, we include

Intr. Rew.	IRM CEM	IRM GD	IRM Rand	Env Roll.	Env CEM	GS	Rand
DADS	83.4 ± 2.19	69.9 ± 2.22	77.2 ± 3.83	74.6 ± 5.15	70.3 ± 5.38	68.9 ± 2.81	28.3 ± 13.5

Table 3: Zero-shot rewards for DADS skill discovery method. IRM CEM substantially outperforms baselines demonstrating generality to skill pretraining methods.

EPIC Loss and Extrinsic Reward are Negatively Correlated

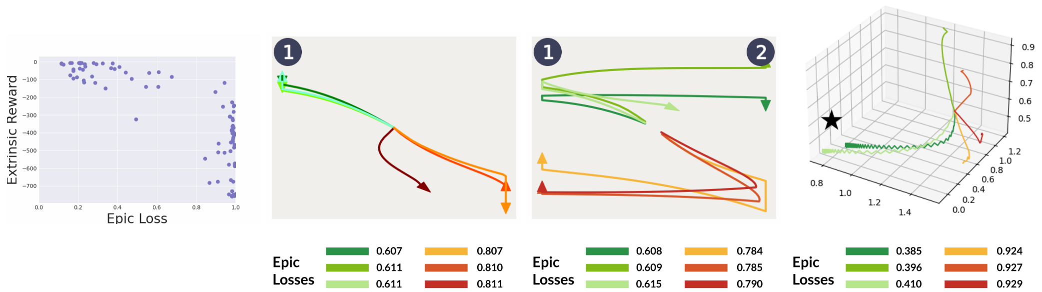


Figure 4: (a) Scatter plot of extrinsic reward vs. EPIC loss, verifying the negative correlation. (b) Trajectories with low and high EPIC losses for planar goal-reaching. (c) Trajectories for sequential goal-reaching. (d) Trajectories for Fetch Reach.

a learned reward scaling parameter for L1 and L2 and similarly observe that EPIC is a superior matching metric.

Skill Discovery Algorithm Ablations IRM is fully general to any mutual information maximization based, RL pretraining algorithm as shown in Table 3. We validate on the Fetch Reach task that IRM CEM and IRM Rand convincingly outperform all episode rollout baselines in zero-shot episode reward.

Reward Matching	IRM CEM
IRM	**20.97** $\pm$ **0.753**
L1	8.71 $\pm$ 0.704
L2	7.87 $\pm$ 0.86
L1 + Learn Scale	5.51 $\pm$ 1.916
L2 + Learn Scale	3.95 $\pm$ 2.217

Table 4: Reward matching metric ablation

5 ANALYSIS

In this section, we provide insight into what the EPIC loss represents and why it serves as a useful and effective proxy for skill selection.

Does optimizing the EPIC loss lead to effective skill selection? Current skill selection relies on evaluating skills based on extrinsic reward obtained through rollouts; can we avoid this and optimize the EPIC loss instead? In Figure 4, we verify that EPIC loss is strongly negatively correlated with extrinsic reward on a Planar Goal Reaching task detailed in A.8. Thus, optimizing for a low EPIC loss is an effective substitute for optimizing the environment reward, and crucially, it forgoes collecting expensive environment samples.

EPIC Loss Visualizations

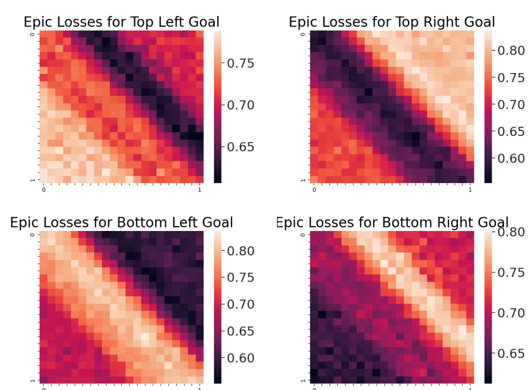


Figure 5: We examine EPIC losses between extrinsic rewards and intrinsic rewards conditioned on the skill vector. We sweep across the 2D skill vector for a pretrained planar agent.

How can we understand skills through EPIC losses? EPIC losses provide insight into the latent structure of skills. In Figure 5, we plot EPIC losses between intrinsic rewards and goal-reaching rewards across the 2D continuous skill space. Not only is there a smoothness to this loss landscape, which motivates optimization methods like gradient descent, but there is also a banded partitioning of the plots. Furthermore, the latent skill space is well-structured as different darker-colored partitions of the skill space correspond to the group of skills with low EPIC loss from each task reward. EPIC losses concisely represent desirability of skills with respect to a downstream reward function, so skills that achieve a low EPIC loss for the Top Left goal will achieve high EPIC losses for the opposite reward, Bottom Right goal.

By utilizing the intrinsic reward module, we can understand the semantics and desirability of a skill in more depth than simply executing the policy and observing extrinsic rewards.

Next, to verify that skills with low EPIC losses are preferable for a corresponding extrinsic reward, we include a scatter plot and trajectory visualizations in Figure 4. As Figure 4 suggests, skills with the lowest EPIC loss receive high extrinsic reward, reaching the goal with high spatial precision. Skills with the highest losses produce the opposite behavior: moving in the direct opposite direction of the goal. In the sequential case, low-EPIC loss skills attempt to reach the 1st goal and 2nd goal; high-EPIC loss skills go in the entirely opposite directions. Furthermore, since reward functions can provide strictly more information regarding the task-level semantics of a policy than a single rolled out trajectory, we regard the EPIC loss as an entirely novel way of interpreting skills and their corresponding semantics.

6 Related Work

Several works including (Sharma et al., 2020) (Eysenbach et al., 2019) (Achiam et al., 2018) (Gregor et al., 2016b) (Baumli et al., 2020) (Florensa et al., 2017) (Laskin et al., 2022) employ mutual information maximization for skill pretraining. While (Laskin et al., 2022) leverages coarse grid search to select skills for downstream RL, methods such as Sharma et al. (2020) instead plan through a learned skill dynamics model at finetuning time. Our approach is similar in that it leverages pretraining model components other than the policy to guide skill selection. However, rather than generating a reward maximizing plan through possibly complex, learned environment dynamics, we instead look to match a policy to the task reward directly through a pretrained discriminator.

In the context of sequential finetuning, (Baumli et al., 2020; Eysenbach et al., 2019) employ hierarchical RL to chain pretrained skills with a meta-controller requiring additional environment interactions. Works on such HRL methods include (Nachum et al., 2018; Frans et al., 2017; Vezhnevets et al., 2017; Springenberg et al., 2018) and more classically (Sutton et al., 1999; Stolle & Precup, 2002). By contrast, we demonstrate that the intrinsic reward matching framework can be extended to choose skill sequences without reliance on environment samples. The successor features line of work also adopts a unified view of skill-based RL. Such work relies on the assumption that arbitrary rewards can be parameterized linearly in some learned features and some task vector as in Liu & Abbeel (2021); Barreto et al. (2016). Our approach relaxes this assumption to the fully general setting by instead searching for a pretrained task with minimal proximity to an arbitrarily parameterized task reward.

7 Discussion

We present Intrinsic Reward Matching (IRM), a framework for algorithmically unifying mutual information maximization unsupervised reinforcement learning with downstream task adaptation. We instantiate a practical algorithm for implementing this framework and demonstrate that IRM performs competitively on continuous control benchmarks and outperforms current methods on more complex tabletop manipulation tasks. IRM diverges from past works in leveraging the discriminator for downstream task inference and consequently performing skill selection without environment interactions in the short horizon setting. We also show that IRM can be readily extended to the general skill sequencing setting to solve more realistic long-horizon tasks as an alternative to hierarchical methods. Central to our contribution is a novel loss function, the EPIC loss, which serves as both a skill selection utility as well as a new way to interpret the task-level semantics of pretrained skills.

We acknowledge a number of limitations of our approach. Principally, our method is heavily reliant on the quality of skill pretraining. Most methods suffer from a form of overfit with larger skill dimensions where the skill discriminator encourages spurious forms of diversity particularly when operating on highly entangled state spaces. Furthermore, the skill policy and skill discriminator are often pretrained with a rather unstable joint optimization. In order for the discriminator to imply an optimal skill policy, the policy must be near optimal for its corresponding intrinsic reward. While we have found that training the policy for more steps can mitigate this misalignment issue, we leave a more thorough investigation for future work.

REFERENCES

Joshua Achiam, Harrison Edwards, Dario Amodei, and Pieter Abbeel. Variational option discovery algorithms. *CoRR*, abs/1807.10299, 2018. URL `http://arxiv.org/abs/1807.10299`.

David Barber and Felix V. Agakov. The im algorithm: A variational approach to information maximization. In *NIPS*, pp. 201–208, 2003. URL `http://papers.nips.cc/paper/2410-information-maximization-in-noisy-channels-a-variational-approach`.

André Barreto, Rémi Munos, Tom Schaul, and David Silver. Successor features for transfer in reinforcement learning. *CoRR*, abs/1606.05312, 2016. URL `http://arxiv.org/abs/1606.05312`.

Kate Baumli, David Warde-Farley, Steven Hansen, and Volodymyr Mnih. Relative variational intrinsic control. *CoRR*, abs/2012.07827, 2020. URL `https://arxiv.org/abs/2012.07827`.

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

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function, 2018.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019.

Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. *CoRR*, abs/1704.03012, 2017. URL `http://arxiv.org/abs/1704.03012`.

Kevin Frans, Jonathan Ho, Xi Chen, Pieter Abbeel, and John Schulman. Meta learning shared hierarchies. *CoRR*, abs/1710.09767, 2017. URL `http://arxiv.org/abs/1710.09767`.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *CoRR*, abs/1802.09477, 2018. URL `http://arxiv.org/abs/1802.09477`.

Adam Gleave, Michael Dennis, Shane Legg, Stuart Russell, and Jan Leike. Quantifying differences in reward functions. *CoRR*, abs/2006.13900, 2020. URL `https://arxiv.org/abs/2006.13900`.

Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control, 2016a.

Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *CoRR*, abs/1611.07507, 2016b. URL `http://arxiv.org/abs/1611.07507`.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *CoRR*, abs/1806.10293, 2018. URL `http://arxiv.org/abs/1806.10293`.

Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale, 2021. URL `https://arxiv.org/abs/2104.08212`.

Michael Laskin, Denis Yarats, Hao Liu, Kimin Lee, Albert Zhan, Kevin Lu, Catherine Cang, Lerrel Pinto, and Pieter Abbeel. URLB: unsupervised reinforcement learning benchmark. *CoRR*, abs/2110.15191, 2021. URL `https://arxiv.org/abs/2110.15191`.

Michael Laskin, Hao Liu, Xue Bin Peng, Denis Yarats, Aravind Rajeswaran, and Pieter Abbeel. Cic: Contrastive intrinsic control for unsupervised skill discovery, 2022. URL `https://arxiv.org/abs/2202.00161`.

Hao Liu and Pieter Abbeel. Aps: Active pretraining with successor features, 2021.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. *CoRR*, abs/1805.08296, 2018. URL http://arxiv.org/abs/1805.08296.

OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving rubik's cube with a robot hand. *CoRR*, abs/1910.07113, 2019. URL http://arxiv.org/abs/1910.07113.

Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A generalist agent, 2022. URL https://arxiv.org/abs/2205.06175.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657*, 2019.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJgLZR4KvH.

David Silver, Aja Huang, Christopher Maddison, Arthur Guez, Laurent Sifre, George Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 01 2016. doi: 10.1038/nature16961.

Jost Tobias Springenberg, Karol Hausman, Martin Riedmiller, Nicolas Heess, and Ziyu Wang. Learning an embedding space for transferable robot skills. 2018.

Martin Stolle and Doina Precup. Learning options in reinforcement learning. In Sven Koenig and Robert C. Holte (eds.), *Abstraction, Reformulation, and Approximation*, pp. 212–223, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

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):181–211, 1999. ISSN 0004-3702. doi: https://doi.org/10.1016/S0004-3702(99)00052-1. URL https://www.sciencedirect.com/science/article/pii/S0004370299000521.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2019.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. *CoRR*, abs/1703.01161, 2017. URL http://arxiv.org/abs/1703.01161.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, L. Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander Sasha Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom Le Paine, Caglar Gulcehre, Ziyun Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy P. Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, pp. 1–5, 2019.

A Appendix

A.1 Background and Notation

Markov Decision Process: The goal of reinforcement learning is to maximize cumulative reward in an uncertain environment it interacts with. The problem can be modelled as a Markov Decision Process (MDP) defined by $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $\mathcal{P}$ is the transition probability distribution, r is the reward function and γ is the discount factor.

Unsupervised Skill Discovery: In competence-based unsupervised RL the aim is to learn skills that generate diverse and useful behaviors Eysenbach et al. (2019). The broad aim is to learn policies that are skill-conditioned and generalizable. Formally, we also learn skills $z \in \mathcal{Z}$ and take actions according to $a \sim \pi(\cdot|s, z)$. As an illustrative example, applying this formalism to the Mujoco Walker domain, we might hope to find a skill-conditioned policy and skills $z_{\text{walk}}, z_{\text{run}}$ such that $\pi(\cdot|s, z_{\text{walk}})$ makes the agent walk, while $\pi(\cdot|s, z_{\text{run}})$ makes it run. Further, if we allow for continuous skills, we can also imagine being able to use the policy to "jog" at different speeds by interpolation the z_{walk} and z_{run} skills. That is, taking $z^{\alpha}_{\text{jog}} = \alpha \cdot z_{\text{walk}} + (1 - \alpha) \cdot z_{\text{run}}$ should, intuitively, yield a policy $\pi(\cdot|s, z^{\alpha}_{\text{jog}})$ that makes the agent jog at speed dictated by the parameter α.

Finetuning Pretrained Skills: With a skill-conditioned policy $\pi(\cdot|s, z)$, an agent needs to infer which skill to index for a downstream task (e.g. identifying if it needs to use z_{walk} or z_{run}) during finetuning. This is a relatively under-explored area, with the most universal approach being a coarse, discretized grid search. Least squares regression has also been investigated in the context of successor features Liu & Abbeel (2021).

A.2 Competence-Based Skill Discovery

Competence-based skill discovery algorithms aim to maximize the mutual information between trajectories and skills:

$$I(\tau; z) = \mathcal{H}(z) - \mathcal{H}(z|\tau) = \mathcal{H}(\tau) - \mathcal{H}(\tau|z) \tag{6}$$

Since the mutual information $I(s; z)$ is intractable to calculate in practice, competence-based methods maximize a variational lower bound. Many mutual information maximization algorithms, such as Variational Intrinsic Control Gregor et al. (2016a) and Diversity is All You Need Eysenbach et al. (2018), use the estimate $I(\tau; z) = \mathcal{H}(z) - \mathcal{H}(z|\tau)$. Other competence-based methods, such as Dynamics-Aware Unsupervised Discovery of Skills Sharma et al. (2019), Active Pretraining with Successor Features Liu & Abbeel (2021), and Contrastive Intrinsic Control (CIC) Laskin et al. (2022), maximize a lower bound for $\mathcal{H}(\tau) - \mathcal{H}(\tau|z)$.

While the decompositions of the mutual information objective are equivalent, algorithms make different design choices regarding how to approximate entropy, represent trajectories, and embed skills. These choices affect the distillation of skills: for instance, without explicit maximization of $\mathcal{H}(\tau)$ in the decomposition of mutual information, behavioral diversity may not be guaranteed when the state space is much larger than the skill space Laskin et al. (2022).

A.3 CIC

Contrastive Intrinsic Control (CIC) Laskin et al. (2022) is a state of the art algorithm for competence-based skill discovery. CIC maximizes a lower bound for $I(\tau; z) = \mathcal{H}(\tau) - \mathcal{H}(\tau|z)$ through a particle estimator for $\mathcal{H}(\tau)$ and a contrastive loss from Contrastive Predictive Coding (CPC) van den Oord et al. (2019) for $\mathcal{H}(\tau|z)$. The lower bound for $I(\tau; z)$ is:

$$I(\tau; z) \geq F_{\text{CIC}}(\tau; z) := \mathcal{H}_{\text{particle}}(\tau_i) + \mathbb{E}\left[q_\phi(\tau_i, z_i) - \log \frac{1}{N} \sum_{j=1}^{N} \exp(q_\phi(\tau_j, z_i))\right] \tag{7}$$

where $\mathcal{H}_{\text{particle}}(\tau) \propto \sum_{i=1}^{n} \log ||h_i - h_i^*||$, h_i^* is the k-Nearest Neighbors embedding, N_k is the number of k-NNs used to approximate entropy, and $N - 1$ is the number of negative samples.

A.4 DADS

We additionally use Dynamics-Aware Unsupervised Discovery of Skills(DADS) Sharma et al. (2020) for skill discovery, as it is one of the few skill discovery algorithms to successfully scale up to continuous skills. DADS maximizes a lower bound for $I(\tau; z) = \mathcal{H}(\tau) - \mathcal{H}(\tau|z)$ through learning skill-conditioned transition distributions. The lower bound for $I(\tau; z)$ is:

$$I(\tau; z) \geq F_{\text{DADS}}(\tau; z) := \log \frac{q_\phi(s'|s, z)}{\sum_{i=1}^{L} q_\phi(s'|s, z_i)} + \log L \tag{8}$$

For our experiments, we reimplement the on-policy DADS algorithm in PyTorch. We follow the default hyperparameters and train for 20 million environment steps, per Sharma et al. (2020).

A.5 Environment Details

The URLB domains are Walker, Quadruped, and Jaco. Walker requires a bipedal agent to perform a variety of navigation based tasks on a 2D-plane while preserving its balance. Quadruped, a more challenging domain due to a higher-dimensional state-action space, requires a quadrupedal agent to perform navigation tasks in a 3D environment. Jaco robot arm is a 6-DOF manipulator with a three-finger gripper which contains a variety of directional reaching tasks

For URLB (Laskin et al., 2021) environments, we follow default environment settings. Like many skill-discovery methods (Sharma et al., 2020) (Eysenbach et al., 2019), we restrict the discriminator input. For Quadruped, we use the x, y, z velocity, which is included in the environment's state space. For Walker, we use the x, y, z world-position, which we add to the environment's state space but remove from the policy input. For Jaco, we use the x, y, z world position.

For our fetch reaching environment, we use the Gym Robotics Fetch environment (Brockman et al., 2016). We set the time limit to 200. For the fetch push environment, we partition the continuous action space into 4 actions, which involve pushing the block forward, backward, left, and right. We set the time limit to 10 for skill learning.

We evaluate sequential skill selection on 2 environments: Fetch Reach and Fetch Push. For the Fetch Push task, we fix 3 waypoints, depicted in Figure 2 and fix a time horizon of 15 pushes per waypoint. For Fetch Reach, we consider 2 waypoints and a time horizon of 25 for each waypoint.

Our plane environment is a 2D world with observations in [-128, 128] x [-128, 128] and continuous actions in [-10, 10] x [-10, 10].

A.6 Pretraining Hyperparameters

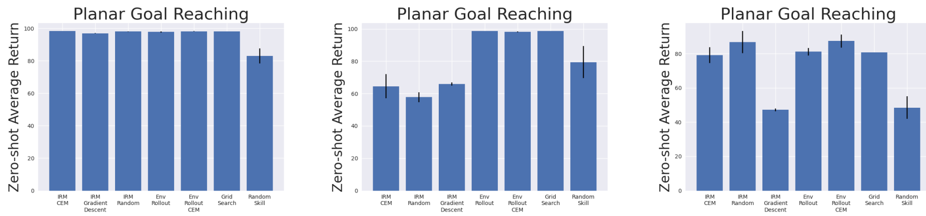
For the Jaco domain we use a skill dimension of 2 and a discriminator MLP hidden dimension of 64. We use an alpha value of 0 for the entropy weighting as in Laskin et al. (2022). We input the 3D position of the end-effector of the Jaco arm to the discriminator. For the Walker domain we use a skill dimension of 2 and a discriminator MLP hidden dimension of 256. We use an alpha value of 0.7 for the entropy weighting. We input the displacement in the 3D position of the torso of the walker to the discriminator. For the Quadruped domain we use a skill dimension of 16 and a discriminator MLP hidden dimension of 128. We use an alpha value of 0.5 for the entropy weighting. We input the 3D velocity of the body of the quadruped to the discriminator. We use a learning rate of 1e-4, a critic target tau parameter of 0.01, and a constant standard deviation exploration schedule of 0.2. The rest of the RL hyperparameters are as in Laskin et al. (2021).

For the Fetch Push environment, we use a skill dimension of 16 and a discriminator MLP hidden dimension of 16. We use an alpha value of 0 for entropy weighting. For the Fetch Reach environment, we use a skill dimension of 8 and a discriminator MLP hidden dimension of 64. We use an alpha value of 0 for entropy weighting. For all environments, we use a replay buffer size of 100k.

A.7 Intrinsic Reward Matching and Environment Rollout Baseline hyperparameters

IRM CEM and Env Rollout CEM are trained for 5 iterations with 1000 samples at each iteration and 100 elites selected each iteration. Env Rollout CEM consumes the entire downstream finetuning

Zero-Shot Performance for Planar Goal Reaching



(a) Skill dimension 2 (b) Skill dimension 8 (c) Skill dimension 16

Figure 6: Zero-Shot Returns for Planar Goal Reaching averaged over 5 seeds

budget on just skill selection. For illustrative purposes, we start its plot at 50k steps to show that finetuning still occurs, however, sample-inefficiency suffers due to excessive rollouts for skill selection. This problem only worsens for long time horizons. IRM Gradient Descent is trained for 5000 steps with a learning rate of 5e-3 and initialized at the skill vector of all 0.5s. IRM Random selects 100 random skills. Env Rollout trials 10 random skills for a fully episode. Grid Search coarsely trials 10 skills from the skill of all 0s to the skill of all 1s as in Laskin et al. (2021).

A.8 Planar Goal Reaching

The planar goal reaching task consists of a simple 2D plane with a point with a 2D Cartesian state space that can displace in the x and y coordinates with a 2D action space. Skills learned tend to span the 2D space reaching to diverse locations distributed broadly across the environment. We show some sample zero-shot skill selection results over three different skill dimensions in 6.

A.9 Finetuning Performance on URLB

In 7 we compare IRM methods against environment rollout-baselines on URLB downstream finetuning sample-efficiency. Since there is less to be gained from precise skill selection due to a limited skill library, and RL methods still require hundreds of thousands of environment steps to learn expert policies, the relative contribution of skipping interaction-based skill identification is not as forthcoming on these tasks. URLB benchmarks tasks with episode lengths between 10 and 25 steps, however, many RL applications call for horizons on the order of many thousands as in (Vinyals et al., 2019). Here, skill-selection algorithms reliant on evaluation rollouts quickly become intractable.

For more scalable rollout based skill selection methods like Env CEM, which can require much more than 100k interactions to converge on a skill, the burden of interaction-reliant skill selection becomes apparent. For illustrative purposes we have shown Env CEM starting at 50k steps even though it far exceeds the 100k sample budget to select a skill before making any RL updates due to having to execute full episode rollouts in the inner loop of optimization. This issue worsens with increasing episode lengths.

A.10 Sequential Skill Selection

For sequential skill selection, we compare IRM Sequential and Environment Sequential skill selection. IRM Sequential consists of an iterative process. The first skill is chosen entirely free of environment samples, exactly identical to the single-skill tasks. Once the first skill is chosen, we roll out a trajectory with the skills we have chosen so far and use the latter half of the trajectory as the Pearson samples for our EPIC loss. We use Gaussian noise with variance 1 for our Canonical samples as described in Appendix A.11.2. At each step of the skill selection process, we opt to select the best next skill through the IRM Random method.

For our Environment Sequential skill selection method, we select skills iteratively as well. For each waypoint or subtask, we randomly sample N skills and commit to the best, where $N = 10/\text{n\_subtasks}$.

Finetuning Performance on URLB

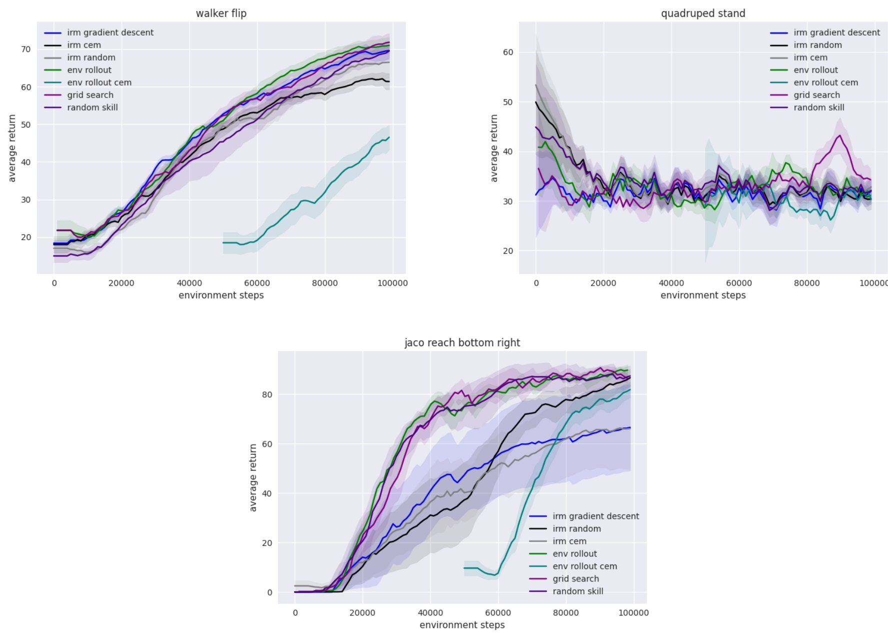


Figure 7: IRM performs competitively on finetuning evaluation with rollout-based methods on denseley rewarded tasks.

Skill Dim	IRM CEM	IRM GD	IRM Rand	Env Roll.	Env CEM	GS	Rand
8	21.1 ± 0.51	15.7 ± 1.61	18.9 ± 0.18	18.4 ± 0.18	18.8 ± 0.48	17.9 ± 0.101	13.5 ± 1.85
16	17.4 ± 1.30	14.6 ± 0.63	18.8 ± 0.26	22.7 ± 0.83	23.1 ± 0.36	14.0 ± 0.19	11.2 ± 2.32
32	20.1 ± 0.54	22.537 ± 0.25	19.8 ± 0.14	22.2 ± 0.58	21.5 ± 0.67	24.0 ± 0.12	19.9 ± 0.67
64	21.9 ± 0.48	1.68 ± 0.069	20.9 ± 0.74	22.5 ± 0.70	21.6 ± 0.89	18.2 ± 0.059	13.3 ± 2.15

Table 5: IRM methods and environment rollout methods ablated over multiple skill dimensions on Fetch Push

A.11 ADDITIONAL ABLATIONS

A.11.1 SKILL DIMENSION

We ablate skill dimension and evaluate the zero-shot performance of all skill selection methods. IRM's performance generally increases with increased skill dimension despite discriminator overfitting issues associated with larger skill spaces. The IRM GD learning rate is chosen as 5e-3 for all experiments in this work and is not tuned at all. Such likely explains the divergence of the 64 dimensional result.

A.11.2 PEARSON & CANONICAL DISTRIBUTIONS

We experiment with many ways to approximate the Pearson and Canonical distributions P_S, P_A, D_S, D_A. In practice, reward functions do not take in actions, so we focus on approximating P_S, D_S.

We defined Full Random to be our uniform samples from a reasonable estimate of the upper and lower bounds for each dimension of the state. For our planar environment, the bounds are defined explicitly and thus known; for more complex environments, we estimate the bounds. In practice, IRM is fairly robust to the distributions, though there are subtleties that emerge in the various choices for the Pearson and Canonical distributions.

We also ablate with a Uniform distribution, which generally performs much worse, due to lack of state coverage for most environments.

For the Canonical distribution, we also approximate samples by perturbing the Pearson samples by ϵ sampled from a Gaussian distribution. We experiment with hyperparameters of variance, which may be adjusted based on the environment. For our sequential IRM method, we use this Canonical distribution to ablate on-policy samples.

All distributions above do not require on-policy samples and thus environment samples, so they are scalable approaches. It is possible to use on-policy samples for P_S, and we choose to do so for our sequential IRM method, as previous skill roll-out may serve as useful Pearson samples for our next skill selection process. Note that while on-policy Canonical samples are possible, they are incredibly expensive and require access into the simulator, so we focus on other distributions of C_S.

Pearson Distribution	**Canonical Distribution**	**IRM CEM**
Full Random	Full Random	20.341 ± 0.306
Full Random	Uniform	16.343 ± 0.708
Full Random	$\epsilon \sim \mathcal{N}(0, 1)$	21.191 ± 0.629
Full Random	$\epsilon \sim \mathcal{N}(0, 0.1)$	21.027 ± 0.419
Uniform	$\epsilon \sim \mathcal{N}(0, 1)$	5.905 ± 3.157
Uniform	$\epsilon \sim \mathcal{N}(0, 0.1)$	2.851 ± 0.605

Table 6: EPIC Loss Sampling Distribution Ablations