Skill-Based Reinforcement Learning with Intrinsic Reward Matching

Anonymous authors
Paper under double-blind review

Abstract

In reinforcement learning (RL), the reward function is a concise yet complete form of task specification. While often used to provide learning supervision to an RL agent, different reward functions can also characterize the varying behaviors in an environment by the optimal policies they induce. In unsupervised reinforcement learning, an agent autonomously learns a family of intrinsic reward functions and corresponding policies with shared latent skill codes. A skill discriminator parametrizes the intrinsic reward function with a neural network where different skill codes correspond to different behaviors. In Intrinsic Reward Matching (IRM), we propose to use this often discarded skill discriminator to understand and use the learned skill policies. Given a downstream task reward function, we use the EPIC reward comparison metric to compare the extrinsic reward function to the skill discriminator-parameterized intrinsic reward function, enabling us to determine which skills correspond to policies that are behaviorally similar to the optimal policys for the new task. We then optimize this metric as a black-box to find the optimal skill and evaluate the skill policy on the downstream task. We demonstrate experimentally that the skill policies IRM selects zero-shot achieve high rewards on the Fetch Tabletop Manipulation and Franka Kitchen domains. Furthermore, we show how IRM can provide insight into the relationships between pretrained skills and downstream tasks.

1 Introduction

Reinforcement learning has seen a rise in multi-task, generalist agents with vast skill repertoires that rival the performance of specialist agents and can even generalize to out-of-distribution tasks (Reed et al., 2022; Kalashnikov et al., 2021). In particular, 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 multi-task RL. During pretraining, unsupervised skill discovery methods learn a discriminator-parameterized family of reward functions corresponding to a family of policies, or skills, through a shared latent code. While prior work has typically focused on learning and composing skills (Sharma et al., 2020; Laskin et al., 2022; Park et al., 2022), we explore how the skill discriminator's intrinsic rewards from pretraining can shed light on the behavioral semantics of the skill policies they induce.

In this work, we present Intrinsic Reward Matching (IRM), an algorithmic methodology for leveraging the learned intrinsic reward function to understand and use pretrained skills. Centrally, we introduce a novel approach to viewing the intrinsic reward model as a multitask reward function that, via zero-shot task inference, enables us to select the optimal pretrained policy for a downstream, extrinsic task reward. Instead of discarding the skill discriminator during finetuning as is often done in prior work (Sharma et al., 2020; Laskin et al., 2022; Park et al., 2022; Gregor et al., 2016; Achiam et al., 2018; Baumli et al., 2020), we discover that the discriminator is an effective task specifier for its corresponding skill policies which can be *matched* with extrinsic reward functions. Our approach views the extrinsic reward as a distribution with measurable proximity to a

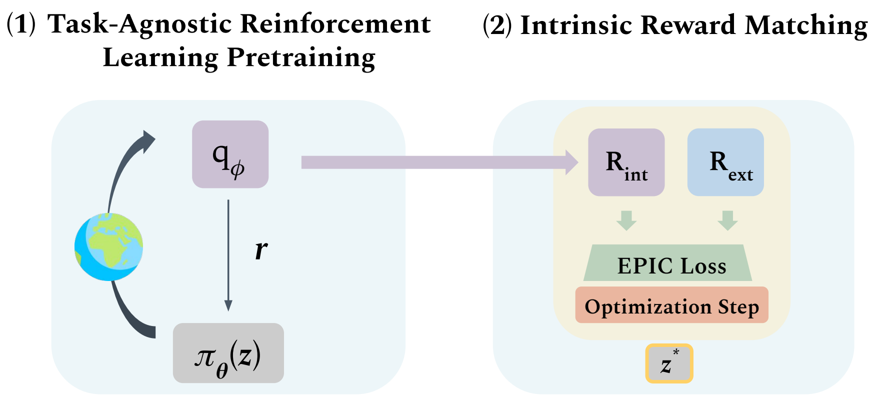
(1) Task-Agnostic Reinforcement Learning Pretraining
(2) Intrinsic Reward Matching
q_ϕ
r
$\pi_\theta(z)$
R_{int}
R_{ext}
EPIC Loss
Optimization Step
z^*

Figure 1: Our proposed method, Intrinsic Reward Matching (IRM), selects an optimal skill that best solves a user-specified reward function. (1) After learning the skills using an off-the-shelf method, (2) our method identifies the most relevant skill based on reward similarity.

pretrained multitask reward distribution and formulates an optimization with respect to skills over a reward distance metric called EPIC (Gleave et al., 2020). We demonstrate experimentally that by minimizing the EPIC distance between the discriminator-parameterized intrinsic reward and a new downstream extrinsic reward as a black box, we can select a skill policy from pretraining that best accomplishes the new task without environment interactions. Furthermore, by analyzing the EPIC loss profile over skill trajectories for different task rewards, we derive insight into the behavioral relationships between pretrained policies and downstream tasks.

2 Background

2.1 Unsupervised Skill Pretraining

The skill learning literature has long sought to design agents that autonomously acquire structured behaviors in new environments (Thrun & Schwartz, 1994; Sutton et al., 1999; Pickett & Barto, 2002). Recent work in competence-based unsupervised RL proposes generic objectives encouraging the discovery of skills representing diverse and useful behaviors (Eysenbach et al., 2019; Sharma et al., 2020; Laskin et al., 2022). A skill is defined as a latent code vector $z \in \mathcal{Z}$ that indexes the conditional policy $\pi(a|s,z)$. In order to learn such a policy, this class of skill pretraining algorithms maximizes the mutual information between sampled skills and their resulting trajectories τ (Gregor et al., 2016; Eysenbach et al., 2019; Sharma et al., 2020) :

$$I(\tau, z) = \mathcal{H}(z) - \mathcal{H}(z|\tau) = \mathcal{H}(\tau) - \mathcal{H}(\tau|z)$$

Commonly, methods consider $\tau = s$ or $\tau = (s, s')$ (Eysenbach et al., 2019; Gregor et al., 2016). We leave more details in Appendix A.2. Since the mutual information $I(s, z)$ is intractable to calculate in practice, competence-based methods instead maximize a variational lower bound proposed in (Barber & Agakov, 2003) which is parameterized by a learned neural network function $q_\phi(\tau, z)$ called a skill discriminator. This discriminator, along with other terms independent of z, parameterizes an intrinsic reward that the skill policy $\pi(\cdot|s, z)$ maximizes during pretraining. Given an unseen task specification, the agent needs to infer which skill produces the most relevant behaviors.

2.2 Pretrained Multitask Reward Functions

We observe 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:

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

where VLB $\leq I(\tau, z)$ is the variational lower bound proposed in (Barber & Agakov, 2003) (τ is a trajectory, e.g. (s, s')). Since skill discovery algorithms maximize $I(\tau, z)$, we view its parameterized lower bound VLB as a multitask reward function: scoring transitions based on alignment with a skill code (Laskin et al., 2022).

2.3 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 distance metric between two reward functions such that similar reward functions induce similar optimal policies. We consider the case of action-independent reward:

$$D_{\text{EPIC}}(R_A, R_B) = \mathbb{E}_{s_P, s'_P \sim D_P, S_C, S'_C \sim D_C}[D_\rho(C(R_A)(s_P, s'_P, S_C, S'_C), C(R_B)(s_P, s'_P, S_C, S'_C))].$$

where $D_\rho(X, Y) = \sqrt{\frac{1-\rho(X,Y)}{2}}$ is the Pearson distance between two random variables X and Y, s_P, s'_P are samples from the Pearson distribution D_P, and S_C, S'_C are batches sampled from the Canonical distribution D_C. We compute the Pearson distance over Pearson samples s_P, s'_P, with additional canonicalization with batches S_c, S'_c to ensure invariance over constant shifts and scaling. The canonicalized reward function is defined as:

$$C(R)(s_P, s'_P, S_C, S'_C) = R(s_P, s'_P) + \mathbb{E}[\gamma R(s'_P, S'_C) - R(s_P, S'_C) - \gamma R(S_C, S'_C)]$$

where $R : S \times S \to \mathbb{R}$ is a reward function and γ is the discount factor (Gleave et al., 2020). The expectation is taken over the Canonical distribution D_C; for simplicity, we sample these batches S_C, $S'_C \sim D_C$ ahead of time. The canonicalization ensures *invariance to reward shaping*, so rewards that have different shaping but induce similar optimal policies are close in distance. In practice, the final term can be omitted as the Pearson correlation is *invariant to constant shifts and scaling.*

The EPIC psuedometric has typically been used for benchmarking algorithms and evaluating learned reward functions (Michaud et al., 2020; Liang et al., 2022). To our knowledge, IRM is the first to use EPIC in an optimization objective, particularly for complex tasks and larger state dimensions.

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 by 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. Thus, we may also expect 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 identifying which of 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 the following optimization:

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

in order to find z^* most aligned with R^{ext}. Moreover, Equation 3.1 performs a minimization of a novel loss we name the *EPIC loss* with respect to the skill z. By EPIC's equivalence class invariance (Gleave et al., 2020), 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 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 For many skill pretraining methods (Laskin et al., 2022; Sharma et al., 2020), the agent requires *negative samples* in order to compute the variational objective in Equation 2.2 and avoid a degenerate optimization where all embedded trajectories have high similarity with all skills. Negative samples correspond to trajectories collected by other skill codes $z' \neq z$. However, during the selection phase when skills are fixed, negative sampling amounts to a constant 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:

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

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.

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 Equation 3.1 as a tractable loss minimization problem.

Canonical State Distribution Approximation: In order to canonicalize our reward functions, we estimate the expectation over the state and next state distributions 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.

Sampling Distribution for Pearson Correlation: We find that generating samples uniformly roughly within the environment workspace bounds, just as with the reward canonicalization, often leads to strong approximations. Furthermore, as both sample generation and relatively inexpensive function evaluation are independent of any online interactions, we can perform the full skill optimization as a self-contained preprocess to downstream policy adaptation without any environment samples. Rough knowledge of workspace bounds represents some amount of prior environment knowledge. We ablate various sampling distribution choices in Table 3 and present the full algorithm in Algorithm 1.

Algorithm 1: Intrinsic Reward Matching (IRM)

```
Require: Downstream task $\mathcal{T}$, $D_P$, $D_C$
Require: Pretrained policy $\pi_\theta(a|s, z)$, intrinsic reward $r_{\text{int}}(s, s', z)$, and extrinsic reward
         $r_{\text{ext}}(s, s')$ for $\mathcal{T}$.
1 while not converged do
2     Sample a batch of Pearson samples $S_P, S'_P \sim D_P, D_P$.
3     Sample Canonical samples $S_C, S'_C \sim D_C, D_C$.
4     Calculate EPIC Loss as $D_{\text{EPIC}}(r_{\text{int}}, r_{\text{ext}}) = D_\rho(C_{D_C}(r_{\text{int}}), C_{D_C}(r_{\text{ext}}))$ (Equation 2.3).
5     Take optimization step on batch with respect to $z$ (Equation 3.1).
6 end while
```

4 Experiments

In this section we experimentally evaluate whether IRM can leverage the intrinsic reward function to identify relevant skills for a downstream reinforcement learning task.

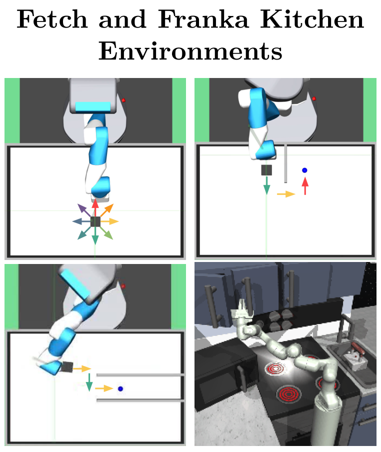

Figure 2: In our Fetch Push environment, we discover skills that move the block in different directions. Downstream tasks may involve simple goals or more distant goals that require composition of multiple skills across an extended time horizon and around obstacles. In Franka Kitchen, the agent discovers skills that interact with various rigid and articulated objects in the scene, which are then leveraged to accomplish diverse tasks.

Environments We design both a goal reaching and a tabletop pushing environment in the OpenAI Gym Fetch environment (Brockman et al., 2016). For the reaching tasks, the robot arm is tasked with reaching towards one (Reach) or many successive (Reach Seq) specified target locations. For the block pushing (Push) environments, we introduce obstacles (Barrier, Tunnel) around which the policy must manipulate the object (Figure 2). Further task design details are provided in Appendix A.5. In addition, we evaluate IRM on 4 manipulation tasks in the Franka Kitchen benchmark (Fu et al., 2020). We use default shaped environment rewards for each task.

Baselines We evaluate 2 variants of IRM that use different optimization methods for minimizing EPIC loss. First, *IRM CEM* uses the Cross Entropy Method (CEM), a Monte-Carlo optimization method, to minimize EPIC loss (described in Appendix A.7). *IRM Gradient Descent* minimizes the EPIC loss using the Adam optimizer to backpropagate through the discriminator to regress the optimal skill.

In addition, we benchmark traditional skill selection approaches. 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 selects the skill which achieves the best empirical reward over an episode (Laskin et al., 2022). *Env Rollout* samples 10 skills uniformly at random from the skill distribution to evaluate with a rollout and chooses the skill with the highest empirical reward (Laskin et al., 2022). *Env Rollout CEM* optimizes the episode reward by using the Cross-Entropy Method (described in Appendix A.7). *Random Skill* selects a skill uniformly at random from the skill distribution to use for the downstream task (Laskin et al., 2021; Eysenbach et al., 2019).

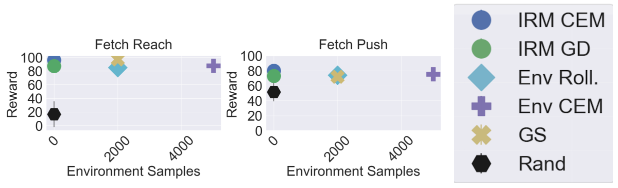

Figure 3: IRM approaches with various optimization methods, even without environment samples, perform favorably in reward compared to environment rollout-based and random skill selection methods.

Evaluation For pretraining skills, we use the Contrastive Intrinsic Control (CIC) (Laskin et al., 2022) algorithm. We use a skill dimension of 8 for Fetch Tabletop Manipulation and a skill dimension of 2 for Franka Kitchen. First, we pretrain each RL agent with the intrinsic rewards for 2M steps. We then evaluate the rewards achieved by the selected skill policy but without any RL updates on the task reward. We report all results averaged over 3 seeds with standard error bars.

4.1 Fetch Tabletop Manipulation and Franka Kitchen

On the Fetch Reach task, IRM outperforms or performs similarly to environment-rollout methods while requiring no environment samples to perform skill selection. As shown in Figure 3, the random skill policy performs poorly and with high variance relative to IRM and environment-rollout based methods. Next, we evaluate IRM on more complex manipulation tasks involving pushing a block to a goal position. This more complex task similarly benefits from bootstrapping the appropriate pretrained skill policy, evidenced by the performance gap of selection based methods over random

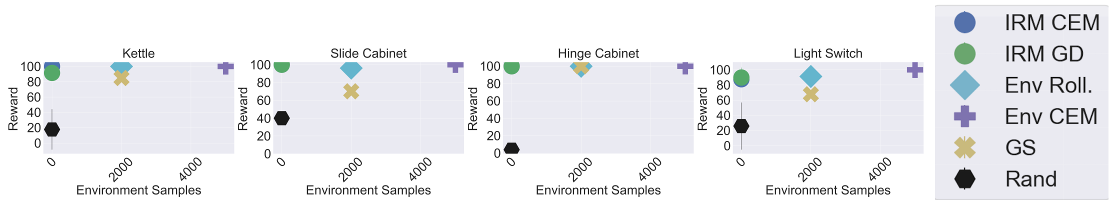

Figure 5: The IRM methods outperform trial-based baselines on Franka Kitchen tasks and without the additional requirement of online environment samples to determine the optimal skill for the downstream reward. When IRM CEM is not visible, note that its value is the same as the IRM GD.

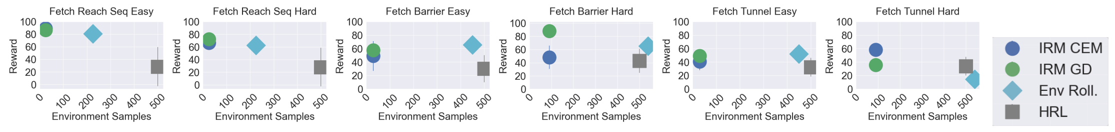

Figure 7: The IRM methods achieve high rewards without environment samples on more complex, long-horizon tasks by finding the skill that matches the intrinsic reward of the discriminator and the task extrinsic reward.

skill selection. Although Env Rollout CEM can be a strong interaction baseline for of zero-shot reward, far exceeding a reasonable budget of interactions entirely on skill selection.

On the Franka Kitchen tasks shown in Figure 5, IRM matches or outperforms the evaluated baselines, while not requiring access to the environment to select an optimal skill. In the case of the Slide Cabinet and Hinge Cabinet tasks, the selected skill is able to fully solve the downstream task zero-shot. For the Light Switch task, IRM performs comparably to the best rollout-dependent baselines, however, does so without any access to the environment, making IRM plausibly more practical for task adaptation when a downstream task reward is provided. Overall, the very low zero-shot performance and often high variance of the random skill policy baseline on this benchmark points to the importance of skill selection methods for adapting pretrained behaviors.

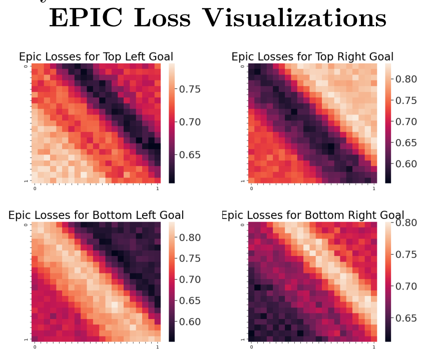

Figure 4: EPIC losses between extrinsic rewards and intrinsic rewards conditioned on the skill. We sweep over the 2D skill vector for a planar agent.

4.2 Skill Sequencing for Long-Horizon Manipulation

Many realistic downstream tasks derive additional complexity from temporally extended planning horizons. We demonstrate that IRM can be used to select sequences of skills to solve longer-horizon tasks with a few assumptions. We consider the long-horizon setting where we have a sequence of reward functions over task horizon H. Central to the skill selection problem is determining over what time intervals should potentially different pretrained skills be selected. In this work, we also predetermine a fixed skill horizon $\lfloor H/N \rfloor$, where N is the number of rewards. However, this skill horizon could also be specified as a parameter and learned from the task reward signal. Next, to perform skill selection over each time interval, we perform the IRM algorithm in parallel or sequentially for each reward. We note the key assumption that IRM requires access to the reward functions for each of the subtasks. For example, for a sequential goal reaching task, we divide the episode into N segments for each of the N goals and corresponding rewards. We perform the IRM skill selection algorithm for each reward to select the optimal skill per interval.

We demonstrate that IRM can be extended to solving long-horizon tasks in the setting of tabletop manipulation. During the unsupervised pretraining phase, skill discovery methods can acquire useful

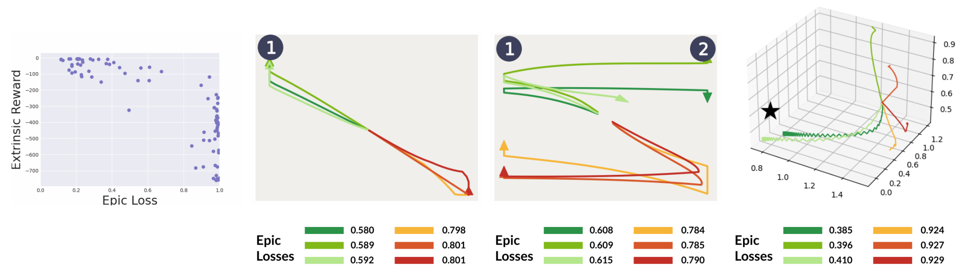

Figure 8: (**left**) Scatter plot of extrinsic reward vs. EPIC loss. (**middle left-middle right**) Skill trajectories with low / high EPIC losses for planar goal-reaching. (**right**) Skill traj. for Fetch Reach.

skills such as directional block pushing or pushing the block to certain spatial locations. We show that IRM can select a sequence of such skills to via reward matching. For the Fetch Reach Sequential tasks, we consider an extended horizon where the agent reaches a sequence of goals in a particular order. For the Fetch barrier tasks, we consider the environments such as depicted in Figure 2, where the agent navigates around a barrier unseen during the pretraining phase. The tunnel environments consists of a goal enclosed in a tunnel; the agent navigates around and into the tunnel to complete the task. We compare IRM methods to an environment rollout baseline, *Env Seq*, and a hierarchical RL baseline, *HRL*, similar to (Eysenbach et al., 2019) in Figure 7. The *IRM Seq* methods select skills based on each defined sub-task's reward function according to the IRM optimization scheme. *Env Seq* chooses the best combination of skills based on extrinsic reward from rollouts. *HRL* initializes a low-level policy with the pretrained skills and optimizes a randomly initialized manager policy over the skill policies. In both experimental settings, IRM methods perform favorably, and crucially, unlike the evaluated baselines, do rely heavily on environment trial-and-error to select a skill. While *HRL* can take advantage of the shaped rewards to choose skills, optimizing another skill selection policy with reinforcement learning requires many more expensive samples and can often be difficult to tune in practice (Eysenbach et al., 2019; Pertsch et al., 2020). Further task and algorithm baseline implementation details are provided in Appendix A.10.

4.3 Analysis and Ablations

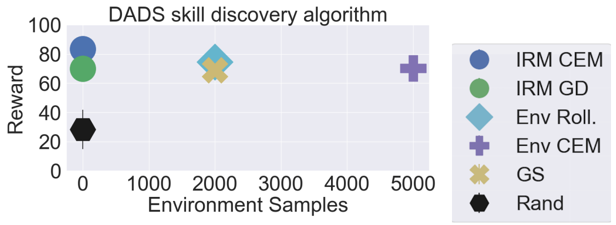

Figure 6: The IRM method outperforms baselines with the DADS skill discovery algorithm, whilst still using no environment interactions to find the best skill.

How Can We Understand IRM Optimization? In Figure 8, we verify that EPIC loss is negatively correlated with extrinsic reward on a Planar Goal Reaching task detailed in Appendix A.9. This provides some evidence that optimizing for a low EPIC loss can lead to higher environment reward. In Figure 4, we plot EPIC losses between intrinsic rewards and goal-reaching rewards across the 2D continuous skill space. Not only is the loss landscape smooth, motivating optimization methods like gradient descent, but there is also a banded partitioning of the manifold. The latent skill space is well-structured: different darker-colored partitions of the skill space correspond to the skills with low EPIC loss from each task reward. In Figure 8, 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 then the 2nd goal, while high-EPIC loss skills perform the behavior in the inverse order. The intrinsic reward module provides deeper insight into the semantics of skills than the extrinsic rewards obtained by skill policy rollouts.

Matching Metric Ablation We validate the importance of employing the EPIC pseudometric by ablating its contribution against more naive selections in Figure 9. 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 a learned reward scaling parameter for L1 and L2 and similarly observe that EPIC is a superior matching metric.

Skill Discovery Algorithm Ablation We show that IRM can work for another popular mutual information maximization, RL skill discovery algorithm, DADS (Sharma et al., 2020). We validate on the Fetch Reach task that IRM CEM and IRM GD match or outperform all episode rollout baselines in zero-shot episode reward as shown in Figure 6.

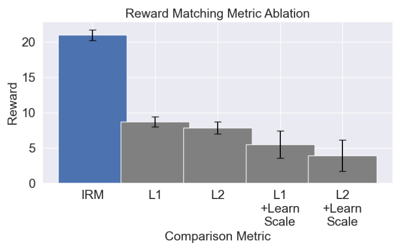

Figure 9: The IRM method is more effective due to the scale and shaping invariances of the EPIC metric.

4.4 Related Work

Several works, including Sharma et al. (2020); Eysenbach et al. (2019); Achiam et al. (2018); Gregor et al. (2016); Baumli et al. (2020); Florensa et al. (2017); Laskin et al. (2022); Campos et al. (2020), employ mutual information maximization for skill pretraining. Recent works, such as Park et al. (2022; 2023a;b); Yang et al. (2023); Zhao et al. (2022), derive alternate intrinsic rewards for skill discovery due to known issues with mutual-information objectives. For skill selection, Laskin et al. (2022) leverages coarse grid search, while Sharma et al. (2020) plan through a learned skill dynamics model during finetuning. Rather than generating reward maximizing plans through possibly complex environment dynamics, we match the policy to task rewards directly through a pretrained discriminator.

Related to zero-shot skill selection is zero-shot reinforcement learning, in which an RL agent can zero-shot solve a task after reward-free pretraining. In Forward-Backward (FB) (Touati et al., 2023; Touati & Ollivier, 2021), learning expected state occupancy can enable quick adaptation to new tasks, and in Successor Features (Barreto et al., 2016) arbitrary rewards can be parameterized linearly in some learned features and a task vector (Liu & Abbeel, 2021; Barreto et al., 2016) enabling task inference to become a linear regression problem. By contrast, our approach searches for a pretrained task with minimal proximity to an arbitrarily parameterized task reward. In addition, zero-shot adaptation using successor features uses Generalized Policy Improvement to extract an improved policy (Barreto et al., 2016), thus requiring access to successor features for previous policies and tasks during inference. By contrast, IRM selects a skill within the existing policy, and does not require access to other policies or rewards during execution.

4.5 Discussion

We present Intrinsic Reward Matching (IRM), a framework for understanding the behaviors of pretrained skills and using them to solve new tasks. We instantiate a practical algorithm for implementing this framework and demonstrate that IRM is effectively able to select pretrained skills for new task rewards without additional environment rollouts. IRM diverges from past works in leveraging the skill discriminator to match the learned family of intrinsic rewards to the new extrinsic task reward. 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 demonstrate experimentally on a variety of tasks that with two different black-box optimizers, we can minimize the EPIC loss to find the best pretrained skill for the downstream reward.

We acknowledge a number of limitations of our approach. IRM relies on samples of the state, roughly within workspace boundaries as well as access to an external reward function, ideally well-shaped, which trades off with IRM's reduced reliance on environment interactions. In order to obtain realistic image samples to compute the EPIC loss, an agent could learn an expressive generative model over the image states obtained during pretraining and sample from the model to generate diverse and realistic sampled states. This represents an interesting direction 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.

Víctor Campos, Alexander Trott, Caiming Xiong, Richard Socher, Xavier Giró-i-Nieto, and Jordi Torres. Explore, discover and learn: Unsupervised discovery of state-covering skills. *CoRR*, abs/2002.03647, 2020. URL https://arxiv.org/abs/2002.03647.

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.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2020.

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. *CoRR*, abs/1611.07507, 2016. URL http://arxiv.org/abs/1611.07507.

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.

Xinran Liang, Katherine Shu, Kimin Lee, and Pieter Abbeel. Reward uncertainty for exploration in preference-based reinforcement learning, 2022. URL https://arxiv.org/abs/2205.12401.

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

Eric J. Michaud, Adam Gleave, and Stuart Russell. Understanding learned reward functions. *CoRR*, abs/2012.05862, 2020. URL https://arxiv.org/abs/2012.05862.

Seohong Park, Jongwook Choi, Jaekyeom Kim, Honglak Lee, and Gunhee Kim. Lipschitz-constrained unsupervised skill discovery, 2022.

Seohong Park, Kimin Lee, Youngwoon Lee, and Pieter Abbeel. Controllability-aware unsupervised skill discovery, 2023a.

Seohong Park, Oleh Rybkin, and Sergey Levine. Metra: Scalable unsupervised rl with metric-aware abstraction, 2023b.

Karl Pertsch, Youngwoon Lee, and Joseph J. Lim. Accelerating reinforcement learning with learned skill priors, 2020.

Marc Pickett and Andrew G Barto. Policyblocks: An algorithm for creating useful macro-actions in reinforcement learning. In *ICML*, volume 19, pp. 506–513, 2002.

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. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJgLZR4KvH.

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.

Sebastian Thrun and Anton Schwartz. Finding structure in reinforcement learning. In G. Tesauro, D. Touretzky, and T. Leen (eds.), *Advances in Neural Information Processing Systems*, volume 7. MIT Press, 1994. URL https://proceedings.neurips.cc/paper/1994/file/7ce3284b743aefde80ffd9aec500e085-Paper.pdf.

Ahmed Touati and Yann Ollivier. Learning one representation to optimize all rewards, 2021.

Ahmed Touati, Jérémy Rapin, and Yann Ollivier. Does zero-shot reinforcement learning exist?, 2023.

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

Rushuai Yang, Chenjia Bai, Hongyi Guo, Siyuan Li, Bin Zhao, Zhen Wang, Peng Liu, and Xuelong Li. Behavior contrastive learning for unsupervised skill discovery, 2023.

Andrew Zhao, Matthieu Gaetan Lin, Yangguang Li, Yong-Jin Liu, and Gao Huang. A mixture of surprises for unsupervised reinforcement learning, 2022.

A Appendix

A.1 Background and Notation

Markov Decision Process: The goal of reinforcement learning is to maximize cumulative reward in an unknown 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.

A.2 Competence-Based Skill Discovery

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

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

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., 2016) and Diversity is All You Need (Eysenbach et al., 2019), 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., 2020), 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:

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

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:

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

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

A.5 Environment Details

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. This is a modification from the original environment, which we implement to ensure more coherent skill learning in a continual action space. We set the time limit to 10 for skill learning.

We evaluate sequential skill selection on Fetch Reach and Fetch Push. For the Fetch Push agent, we have a Barrier and Tunnel environment, each with 3 waypoints, depicted in Figure 2. We fix a time horizon of 30 pushes per waypoint. For Fetch Barrier, the easy task consists of the first two waypoints (L shape), and the hard task consists of all three (U shape). For Fetch Tunnel, the easy task consists of the last two (into the tunnel), and the hard task consists of all three waypoints (side of the tunnel, and then into the tunnel). For Fetch Reach, we consider 2 waypoints and a time horizon of 25 for each waypoint. The extrinsic reward function for Fetch Reach and Push tasks are negative L2 distances from the goals.

Our plane environment is a 2D world with observations in [-128, 128] x [-128, 128] and continuous actions in [-10, 10] x [-10, 10]. The extrinsic reward function is the negative L2 distance from the goal.

The Franka Kitchen Environment is a standard benchmark consisting of a 9 dof Franka Arm in a Kitchen Environment with a variety of rigid and articulated objects. The benchmark specifies several goals with rewards for reaching those goals. The 7d action space includes the end-effector 3d translation, 3d rotation, and a gripper open/close dimension. The state includes the joint positions corresponding to each dof of the arm in addition to the joint positions of articulated bodies and 7-dimensional position and orientation information for rigid objects in the scene. The Kettle task involves moving the kettle to a target burner location on the stove. The Hinge and Slide Cabinet tasks involve opening cabinet drawers with revolute and prismatic joints respectively. The Light Switch task involves flipping the switch for the stove light.

The Franka kitchen rewards are negative L2 distances to goals. For the Kettle task, the goal is a translational and rotational space position goal. For the hinge cabinet task, it is a target rotational angle. For the slide cabinet task, it is a target prismatic joint position. For the light switch task, it is a target rotation angle. The latter three task rewards are somewhat sparse because there is no negative L2 distance reward in 3D Cartesian space for reaching the correct articulated object.

A.6 Pretraining Hyperparameters

We include hyperparameters in Table 1. For the on-policy DADs algorithm, we do not keep any off-policy samples in a replay buffer.

A.7 Cross-Entropy Method (CEM)

In IRM CEM and Env Rollout CEM, we use the Cross-Entropy Method, a Monte-Carlo method for optimization. The Cross-Entropy Method consists of iterating between two phases: (1) Sampling from a probability distribution (2) Fitting a probability distribution on the drawn samples. In our

	Fetch Reach	Franka Kitchen
Skill Dim	8	2
Discriminator MLP Hidden Dim	64	64
α (CIC entropy weight)	0	0.1
Replay Buffer Size	100k	100k

Table 1: Pretraining Hyperparameters.

implementation of CEM, we draw a batch of samples (hyperparameters listed in the section below) and select the number of elites, which are the samples with the ideal optimization value (i.e. the lowest EPIC loss or highest environment reward). Then, we fit the mean and standard deviation of a Gaussian to the domain of the elites (i.e. mean and variance of sampled skills). Then, we repeat the process of sampling skills from the updated distribution, selecting elites, and refitting the probability distribution.

A.8 Matching Algorithm 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 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.9 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.

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. For our Fetch Reach environments, each skill is rolled out for 25 steps, and for Fetch Barrier and Tunnel environments, each skill is rolled out for 30 steps. For instance, Fetch Barrier Hard and Fetch Tunnel Hard consist of tasks with 3 sequential skills: thus, during skill selection, the first skill is selected without any environment steps, the second skill is selected by rolling out the first skill (30 environment steps), and the third skill is selected by rolling out the first two skills (60 steps). Thus, for our sequential skill selection tasks, this amounts to 25 environment steps in Fetch Reach Seq Easy and Hard; 30 environment steps for Fetch Barrier Easy; $30 + 60 = 90$ steps for Fetch Barrier Hard; 30 steps for Fetch Tunnel Easy; $30 + 60 = 90$ steps for Fetch Tunnel Hard.

We use Gaussian noise with variance 1 for our Canonical samples as described in Appendix A.13. At each step of the skill selection process, we use the corresponding IRM optimization methods.

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 $N = \lfloor 10/\text{n\_subtasks} \rfloor$. Per each subtask, we roll out the selected skills from the previous subtasks (if there are previous subtasks), and then roll out N different skills. Concretely, for Fetch Reach Seq

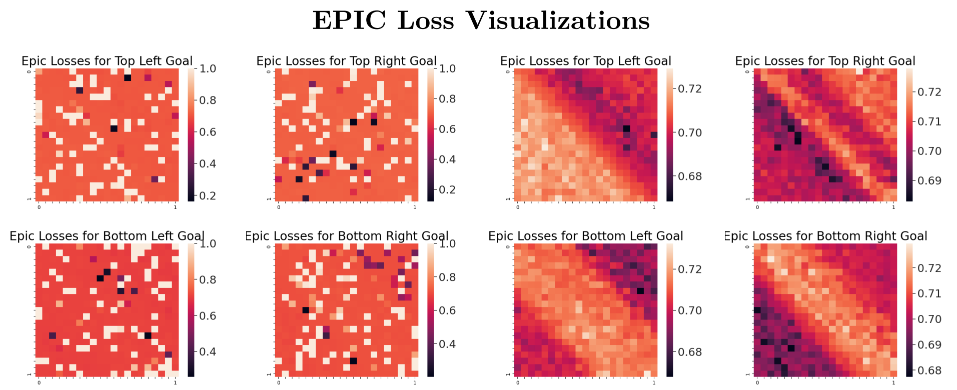

Figure 10: We examine EPIC losses between extrinsic rewards and intrinsic rewards conditioned on the skill vector. We sweep across the 2D skill vector for a pretrained planar agent. Left: Sparse goal-reaching reward with tolerance of 0.03. Right: Sparse goal-reaching reward with tolerance of of 0.07.

Skill Dim	IRM CEM	IRM GD	Env Roll.	Env CEM	GS	Rand
8	21.1 ± 0.51	15.7 ± 1.61	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	22.7 ± 0.83	23.1 ± 0.36	14.0 ± 0.19	11.2 ± 2.32
32	20.1 ± 0.54	22.5 ± 0.25	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	22.5 ± 0.70	21.6 ± 0.89	18.2 ± 0.059	13.3 ± 2.15

Table 2: IRM methods and environment rollout methods ablated over multiple skill dimensions on Fetch Push

Easy and Hard, there are a total of 25 * 3 = 75 environment steps for selecting the first skill, and (25 + 25) * 3 = 150 environment steps for selecting the second skill, leading to a total of 225 steps. For Fetch Barrier Easy, $N = 5$, and there is a total of 30 * 5 + (30 + 30) * 5 = 450 steps. For Fetch Barrier Hard, $N = 3$, and there is a total of 30 * 3 + (30 + 30) * 3 + (30 + 30 + 30) * 3 = 540 steps. For Fetch Tunnel Easy, $N = 5$, and there is a total of 30 * 5 + (30 + 30) * 5 = 450 steps. For Fetch Tunnel Hard, $N = 3$, and there is a total of 30 * 3 + (30 + 30) * 3 + (30 + 30 + 30) * 3 = 540 steps.

A.11 Sparse Reward Ablation

We ablate our planar EPIC Loss visualizations with sparse rewards. Instead of a well-shaped goal-reaching reward, we use sparse rewards based on the tolerance to the goal. We define the tolerance as the radius the agent must be within if our 2d planar environment is scaled to [0, 1] x [0, 1]. With a very sparse reward, we show that EPIC losses are largely uninformative. However, by slightly relaxing the tolerance, we show a much better shaped EPIC loss landscape that bears similarity to that of Figure 4. Thus, while our method is dependent on access to extrinsic rewards, and ideally, shaped rewards, we show that the EPIC loss landscape over sparse reward landscapes with sufficient tolerance can be optimized.

A.12 Skill Dimension Ablation

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.13 Pearson & Canonical Distribution Ablation

We experiment with many ways to approximate the Pearson and Canonical distributions. 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. For example, for a tabletop manipulation workspace, we sample 2-dimensional block positions uniformly within the rectangular plane of the table surface. 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. For instance, we also ablate a Uniform(0,1) 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.

We leave more general options such as sampling from a learned generative model over trajectories encountered during pretraining or sampling from saved pretraining data to future work.

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

Table 3: EPIC Loss Sampling Distribution Ablations.

None of the distributions ablated above require on-policy environment samples. It *is* possible to use on-policy samples for the state distributions, and we choose to do so for our sequential IRM method, as previous skill rollouts may provide useful Pearson samples for the subsequent skill selection. Note that while on-policy Canonical samples are possible, they are incredibly expensive and require access to the environment simulator, so we focus on other choices of distributions.

Task	**IRM CEM**	**IRM GD**	**Env Roll.**	**Env CEM**	**GS**	**Rand**
Fetch Reach	95.9 ± 1.0	87.5 ± 0.20	85.0 ± 6.2	87.8 ± 1.9	**97.3 ± 0.00**	16.7 ± 19
Fetch Push	**80.2 ± 2.5**	73.1 ± 0.48	74.3 ± 0.92	75.4 ± 2.6	72.1 ± 0.00	51.5 ± 12.5

Table 4: IRM with various optimization methods compared to environment rollout-based skill selection and random skill selection. IRM based methods rival or exceed skill selection baselines that are reliant on expensive environment trials.

Task	**IRM CEM**	**IRM GD**	**Env Roll.**	**HRL**
Fetch Reach Seq Easy	**89.5 ± 0.34**	86.7 ± 0.64	80.7 ± 4.7	28.4 ± 31.0
Fetch Reach Seq Hard	66.4 ± 1.1	**72.4** ± 3.3	62.7 ± 6.1	28.0 ± 30.6
Fetch Barrier Easy	49.0 ± 22.1	**57.5 ± 9.5**	65.5 ± 9.0	30.1 ± 20.0
Fetch Barrier Hard	48.1 ± 17.8	**88.2 ± 10.4**	65.4 ± 5.6	42.8 ± 18.2
Fetch Tunnel Easy	40.8 ± 13.0	49.1 ± 8.4	**52.2 ± 9.1**	32.2 ± 13.9
Fetch Tunnel Hard	**58.4 ± 1.8**	35.7 ± 1.3	14.5 ± 4.8	33.7 ± 13.9

Table 5: Rewards on long-horizon manipulation tasks

Task	**IRM CEM**	**IRM GD**	**Env Roll.**	**Env CEM**	**GS**	**Rand**
Kettle	**105** ± 0.00	96.2 ± 6.3	105 ± 0.27	105 ± 0.00	89.1 ± 0.00	18.7 ± 26.4
Slide Cabinet	**5.00** ± 0.00	5.00 ± 0.00	4.81 ± 0.13	5.00 ± 0.00	3.51 ± 0.00	2.00 ± 0.34
Hinge Cabinet	**175** ± 0.00	175 ± 0.00	175 ± 0.00	175 ± 0.00	175 ± 0.00	7.02 ± 75
Light Switch	182 ± 9.3	187 ± 12.9	190 ± 13.4	**208** ± 0.32	141 ± 0.00	53.7 ± 31

Table 6: Rewards for Franka Kitchen tasks