LEARNING SKILLS DIVERSE IN VALUE-RELEVANT FEATURES

Matthew J.A. Smith[‡] [*]
University of Oxford

Jelena Luketina[‡] [†]
University of Oxford

Kristian Hartikainen
University of Oxford

Maximilian Igl
University of Oxford

Shimon Whiteson
University of Oxford

ABSTRACT

Behavioural abstraction via temporally extended actions is vital to solving large-scale reinforcement learning problems. Skills structure exploration, speed up credit assignment, and can be used in transfer learning. However, such abstraction is often difficult or expensive for experts to craft by hand. Unsupervised information-theoretic methods (Gregor et al., 2016; Eysenbach et al., 2019; Sharma et al., 2020) address this problem by learning a set of skills without using environment rewards, typically by maximizing discriminability of the states visited by individual skills. However, since only some features of the state matter in complex environments, these methods often discover behaviours that are trivially diverse, learning skills that are not helpful for downstream tasks. To overcome this limitation, we propose a method for learning skills that only control features important to the tasks of interest. First, by training on a small set of source tasks, the agent learns which features are most relevant. Then, the discriminability objective for an unsupervised information-theoretic method is defined for this learned feature space. This allows the construction of sets of diverse and useful skills that can control the most important features. Experimental results in continuous control domains validate our method, demonstrating that it yields skills that substantially improve learning on downstream locomotion tasks with sparse rewards.

1 INTRODUCTION

The automatic learning of useful skills is a fundamental challenge in scaling reinforcement learning (RL). Such behavioural abstraction can reduce the effective time horizon of RL problems and enable more coordinated exploration (Dietterich, 1998; Nachum et al., 2019). Furthermore, if skills are modular, they can facilitate transfer learning, as each skill constitutes a part of a solution (Andreas et al., 2017; Frans et al., 2018; Igl et al., 2019b).

Rather than specify skills manually, which can be difficult or expensive, many existing approaches look to learn them automatically. In particular, one family of algorithms learns without use of a reward function, yielding skills that can be used across a variety of downstream tasks. These recent works (Gregor et al., 2016; Eysenbach et al., 2019; Sharma et al., 2020) learn distinct skills that are optimized such that trajectories generated by different skills are maximally distinguishable, according to the states visited by each skill. This yields a coordinated optimisation problem: a discriminator network is optimized to distinguish skills, and skill policies are optimized to be distinct. However, skills learned in this way do not always capture meaningful structure in the environment, as in rich state spaces, the discriminability objective may have many optima, leading to many different ways in which skills could differ.

In practice, without explicit encoding of prior knowledge about the specific features with respect to which skills should be diverse, the learned skills are often trivial. For example, agents may learn subtle manipulations of state that are imperceptible to humans, or subtly change joint angles in MuJoCo tasks (Achiam et al., 2018; Eysenbach et al., 2019). These skills, while perfectly distinguishable from each other, do not manipulate the state in ways useful for typical tasks of interest. This problem becomes acute when the state space is large and some features are easily controllable. As an extreme example, imagine a cooking agent that employs visual input as state. In this setting, we cannot explicitly express the temperature or spiciness of a dish in pixel space. This prohibits the use of existing methods, as without access to hand-crafted features, the agent would learn to manipulate complex state spaces in easy-to-learn but unhelpful ways (e.g., knocking all the spices from the shelf, but adding none to the dish). Instead, if the agent specifically learns

[*] Contact: msmith@cs.ox.ac.uk
[†] Contact: jelena.luketina@cs.ox.ac.uk
[‡] Denotes Equal Contribution Author

to manipulate features of the room that determine the spiciness and temperature of the dish being prepared, then the agent will be more likely to learn to prepare many dishes quickly.

To achieve this, we propose DIVRS (Diverse In Value Relevant Space), a semi-supervised, sequential method that bridges the gap between hand specification and unsupervised learning of behavioural abstraction. Our framework uses learning to make it easier to specify the features of the environment in which skills should be differentiated. To do so, we first make use of a small amount of reward signal to learn a representation of state that enables prediction of task performance. Then, we learn skills that manipulate the space of those relevant and controllable features. We demonstrate that, compared to purely unsupervised counterparts, such skills are better able to cover the space of possible value functions that depend on those relevant features. Such skills can then be utilized on more challenging downstream tasks that share those same features.

2 PRELIMINARIES

We define a discounted Markov Decision Process (MDP) as a tuple $M = \langle S, A, P_0, T, r, \gamma \rangle$ where $S \subseteq \mathbb{R}^{N_s}$ is the set of states, A the set of actions, T the transition probability function $T : S \times A \times S \to [0, 1]$, r the reward function $r : S \times A \to \mathbb{R}$, P_0 the initial state distribution $P_0 : S \to [0, 1]$, and $\gamma \in [0, 1)$ is a discount factor. The objective is to find a parametric policy $\pi_\theta(a|s) = p(A = a|S = s)$ that maximizes the expected discounted cumulative return $\mathcal{J}(\theta) = \mathbb{E}_\tau[\sum_{t=0}^{\infty} \gamma^t r(a_t, s_t)]$, where $\tau = \{s_0, a_0, s_1, ...\}$ is a trajectory generated under π_θ: $s_0 \sim P_0$, $a_t \sim \pi_\theta(a_t|s_t)$, $s_{t+1} \sim T(s_{t+1}|s_t, a_t)$.

2.1 PROBLEM SETTING

We are given access to an environment $\langle S, A, T, P_0 \rangle$ that can be freely explored. The environment also consists of a set of tasks $\mathcal{M} = \{M_i\}_{i=1...N_m}$, with each task M_i specified by $\langle r_i, \gamma_i \rangle$. Our goal is to pre-train a set of skills $\pi(a|s, z)$, $z \in \{1, 2, \ldots, N_z\}$ such that any of the tasks M_i can be solved quickly. For example, when we anticipate having to train multiple agents on different tasks from $\mathcal{M}$, we can reduce overall training costs by pre-training a set of useful skills only once.

During pre-training, we use a small subset of representative *source* tasks $\mathcal{M}_s \subset \mathcal{M}$. In general these tasks can be randomly sampled from the task distribution, or chosen by hand to be easy to solve, or represent well the space of tasks in $\mathcal{M}$ well. As discussed further in Section 3, access to $\mathcal{M}_s$ allows us to learn skills that are relevant for $\mathcal{M}$, without having to directly solve the large number of tasks in $\mathcal{M}$ individually, or solve particularly difficult tasks in $\mathcal{M}$. In fact, the source tasks do not even need to be fully solved; only effective policy evaluation is needed to learn useful skills. For this to be feasible, we assume the existence of a shared structure between the tasks. Specifically, we assume there is a set of low-dimensional shared state features $\psi(s)$ with $\dim(\psi(s)) << N_s$, that are useful for predicting the long-term performance on all of the tasks from $\mathcal{M}$, though in general, these features may be difficult to learn.

Such assumptions hold in many real-world problems. In navigation tasks, for example, the value of a state is primarily determined by the agent's location or its distance from particular objects (with the same feature potentially having positive or negative value depending on the task instance). For a cooking robot tasked with making many different dishes, the features relevant for differentiating the dishes include the amounts of ingredients, the temperature under which it was cooked, the spiciness or acidity of the dish, etc.

2.2 VARIATIONAL OPTION DISCOVERY

Variational option discovery (VOD) methods (Gregor et al., 2016; Eysenbach et al., 2019; Achiam et al., 2018) are a recently proposed family of techniques for unsupervised skill discovery. A skill π_z is defined as a policy $\pi(a|s, z)$ conditioned on a discrete latent variable z. VOD algorithms aim to discover a set of diverse skills such that each skill can be uniquely identified from some statistic ξ_τ derived from a trajectory τ generated by rolling out that skill policy $\pi_\theta(a|s, z)$ for an entire episode. Typical choices for ξ_τ are a subsequence of visited states $\{s_0, s_k, s_{2k}...\}$ as in VALOR (Achiam et al., 2018), individual states s_t as in DIAYN (Eysenbach et al., 2019) or terminal states as in VIC (Gregor et al., 2016). Formally, the skills are trained to maximize the mutual information (MI) between z and the statistic ξ_τ, $I(z; \xi_\tau)$, sometimes including maximization of the entropy $H[\pi_\theta(a|s, z)]$ to help ensure good exploration.

Due to the symmetry of MI, two decompositions are generally employed:

$$I(z; \xi_\tau) = H(z) - H(z|\xi_\tau) \tag{1}$$

$$= H(\xi_\tau) - H(\xi_\tau|z), \tag{2}$$

(1) is often called the *reverse* MI and (2) the *forward* MI (Gregor et al., 2016; Campos et al., 2020). Optimizing either form directly is generally intractable. Fortunately, the reverse form admits maximization of a variational lower bound instead, resulting in the entropy-regularised objective:

$$\max_{\theta,\phi} \mathbb{E}_{z\sim P(z)}[\mathbb{E}_{\tau\sim\pi_z,T,P_0}[\log q_\phi(z|\xi_\tau) + \beta H(\pi_\theta(a|s,z))] - \log P(z)], \tag{3}$$

By contrast, the forward objective can only be bounded approximately, due to the intractability of both $\log P(\xi_\tau|z)$ and $\log P(\xi_\tau)$, which have opposite signs. This approximation (with entropy regularisation) is given by:

$$\max_{\theta,\phi} \mathbb{E}_{z\sim P(z)}[\mathbb{E}_{\tau\sim\pi_z,T,P_0}[\log q_\phi(\xi_\tau|z) - \log \sum_{z'} q_\phi(\xi_\tau|z')P(z') + \beta H[\pi_\theta(a|s,z)]]], \tag{4}$$

Despite this, the forward objective has been used effectively (Sharma et al., 2020; Campos et al., 2020), indicating perhaps that the powerful variational models applied are able to closely fit the true distribution.

In both cases, the latent distribution $P(z)$ is usually not learned and set to be uniform in order to ensure that all skills are trained equally. The training procedure consists of two simultaneous optimization processes: the variational model, referred to as the *discriminator*, is fit to minimize the negative log-likelihood loss $-\log q_\phi$; while a skill-conditional policy maximizes the corresponding lower bound in either (3) or (4). In other words, for the reverse objective, the discriminator is trained to predict skills from the trajectories, while for the forward one it is trained to predict the states that are visited by the skills. In both cases policies are rewarded for producing trajectories on which the discriminator performs well. In practice, skills learned via forward MI tend to be unimodal, since each skill is specified by a single state, while reverse mode skills are often multimodal (Campos et al., 2020).

For sufficiently large state spaces in which the agent is able to easily control the state, diversity in the raw state space is not sufficient to learn skills that are relevant for downstream tasks. For example, the set of skills discovered by a MuJoCo Ant agent trained with DIAYN (Eysenbach et al., 2019) tend to take actions that affect easily-controllable features like joint positions and orientation. While these skills are distinguishable, they do not capture meaningful behaviours such as different walking gaits or even involve significant or consistent movement in the x-y plane, rendering them useless for most tasks, including navigation. The underlying problem is that some features of the state, like joint angles, are much more directly influenced by actions and thereby easier to control by the policy than others, like the location of the agent, which requires a longer sequence of coordinated actions to change.

To find skills that are useful for navigation, a common approach has been limiting the observation space of the discriminator to the position of the agent's centre of mass (referred to as an x-y prior) (Eysenbach et al., 2019; Sharma et al., 2020) or the difference between consecutive states (Achiam et al., 2018) to encourage skills that exhibit movement. However, there are many tasks in which the relevant features of the state space are unknown a priori or are hard to specify in this manner.

3 METHOD

To address these problems, we propose DIVRS (Diverse In Value Relevant Space), which utilizes the source tasks in $\mathcal{M}_s$ to extract a feature mapping $\psi_\omega(s) : S \to S^\psi$ that captures only those variations in the state that matter for the tasks of interest. By then restricting the discriminator's observations to such a latent representation, we learn skills that are not only diverse, but diverse in relevant features. DIVRS consists of two phases of learning. The aim of the first phase is to learn a set of good state features, using the small set of source representative tasks $M_i \in \mathcal{M}_s$. After learning good features for the source tasks, we need to extract the representation $\psi_\omega(s)$ from this agent. In the second step, we learn skills that reach states that are *diverse* in those features. The pseudo-code for both phases can be found in Algorithm 1, with the notation explained in the rest of this section.

3.1 PHASE ONE

Choice of features In order to capture features that are relevant specifically on the tasks of interest, we propose that $\psi_\omega(s)$ be given by features of the value function that are shared across tasks. Since the value function captures the long-term performance of the agent, the features that it depends on describe exactly how successful the agent is likely to be—if the agent is seeing features that predict high values, it is likely to perform well on the given task. By using the state features of the value function, we can obtain skills that manipulate ψ_ω in a way that maximizes a wide range of value functions in $\{\mathcal{M}_i\}$. We choose features of the value function over a learned reward model because the value function is smooth compared to the reward function and carries more information about the long term effect of various features of the environment. Note that, while there are many other approaches one could use to learn features in

RL, including autoencoders (Lange & Riedmiller, 2010), and successor features (Schaul et al., 2015), these methods generally do not depend on task perfomance, which is our focus here.

Algorithm 1 DIVRS

Input: set of source tasks $\{\mathcal{M}_1, ..., \mathcal{M}_k\}$, coverage policy μ
Output: set of skills $\pi_\theta(a|s, z)$

Phase 1 - Learning State Representation $\psi_\omega(s)$

initialize $\psi_{\hat{\omega}}(s), \{w_1, ..., w_k\}, f_v$
while $\psi_{\hat{\omega}}(s)$ is not converged **do**
  sample $\mathcal{M}_i$ from $\{\mathcal{M}_1, ..., \mathcal{M}_k\}$
  for j in n_steps **do**
    transition_batch $\leftarrow$ rollout_policy($\mathcal{M}_i$, μ)
    $\psi_{\hat{\omega}}(s), \{w_i\}, f_v \leftarrow$ **PEval**(transition_batch, $\psi_{\hat{\omega}}(s), \{w_i\}, f_v$)
$\psi_\omega(s) \leftarrow$ **Distillation**($\{\mathcal{M}_1, ..., \mathcal{M}_k\}, \psi_{\hat{\omega}}(s), \mu$)

Phase 2 - Learning Skills $\pi_\theta(a|s, z)$

fix the parameters ω of $\psi_\omega(s)$
initialize $\pi_\theta(a|s, z), q_\phi(z|\psi_\omega(s)), Q_\nu(s, a|z)$
while not converged **do**
  sample $z \sim P(z)$
  for j in n_steps **do**
    exps $\leftarrow$ rollout_policy($\pi_\theta(a|s, z)$)
    exps.reward $\leftarrow \log q_\phi(z|\psi_\omega(s_t)) - \log P(z)$
    $\pi_\theta(a|s, z), Q_\nu(s, a|z) \leftarrow$ SAC_update(exps, $\pi_\theta(a|s, z), Q_\nu(s, a|z)$)
    $q_\phi(z|\psi_\omega(s)) \leftarrow$ optimize_dscr_loss(exps, $q_\phi(z|\psi_\omega(s))$)

Algorithm 2 PEval

Input: transition batch D, shared embedding $\psi_{\hat{\omega}}(s)$, task embedding w, value network f_v

$S, A, R, S' \leftarrow D$
$\delta \leftarrow (f(\psi_{\hat{\omega}}(S), w) - R - \gamma f_v(\psi_{\hat{\omega}}(S'), w))$
$v \leftarrow v - \delta \nabla_v f_v(\psi_{\hat{\omega}}(S), w)$
$\hat{\omega} \leftarrow \hat{\omega} - \delta \nabla_{\hat{\omega}} f_v(\psi_{\hat{\omega}}(S), w)$
$w \leftarrow w - \delta \nabla_w f_v(\psi_{\hat{\omega}}(S), w)$
return $\psi_{\hat{\omega}}(s)$, w, f_v

Algorithm 3 Distillation

Input: A set of source tasks $\{\mathcal{M}_1, ..., \mathcal{M}_k\}$, target embedding $\psi_{\hat{\omega}}(S)$, coverage policy μ
Output: Distilled Embedding $\psi_\omega(S)$

initialize bottleneck network ω, reconstruction network d_z
while $\|\psi_{\hat{\omega}}(s) - d_z(\psi_\omega(s))\| \geq \epsilon$ **do**
  sample $\mathcal{M}_i$ from $\{\mathcal{M}_1, ..., \mathcal{M}_k\}$
  for j in n_steps **do**
    $\leftarrow$ rollout_policy($\mathcal{M}_i$, μ)
    $l \leftarrow$ RegressionLoss($d_z(\psi_\omega(S)), \psi_{\hat{\omega}}(S)$)
    $z \leftarrow z - \nabla_z l$
    $\omega \leftarrow \omega - \nabla_\omega l$
return $\psi_\omega(s)$

Learning features To learn value-relevant features, we fix a coverage policy μ across all tasks for the first phase and simply perform policy evaluation, while ensuring that $\psi_\omega(s)$ is optimized across all tasks. This policy could be random, derived from expert behaviours, or chosen to cover state space uniformly. Alternatively, the policy can be conditioned on the active task, and optimised alongside the value function via policy improvement. In our experiments, random policies were sufficient to derive good skills. Due to the variance inherent in evaluation of random policies, we found that it was easier to first train a value network with a high dimensional embedding, $\psi_{\hat{\omega}}(s)$, which is later compressed in a distillation phase, which we describe below. The training procedure for $\psi_{\hat{\omega}}(s)$ is detailed in Algorithm 1 (Phase 1), and Algorithm 2. Since DIVRS primarily relies on policy evaluation, optimal policies do not need to be learned for the source tasks. Instead, the value function only needs to capture the features that are predictive of future rewards.

Value function architecture Crucially, for DIVRS to work, the *reward on the source tasks does not have to be dense* as long as policy evaluation can be performed with good coverage of states that lead to rewards. To ensure that $\psi_{\hat{\omega}}$ remains task-independent and is used across all tasks, we can choose our class of value function estimators V_i^μ (as in (Schaul et al., 2015)) such that values can be factored as a combination of a state-independent task embedding and the learned features, i.e. $V_i^\mu(s) \approx f(\psi_{\omega(s)}, w_i)$, where $f(\cdot)$ is a simple function like inner product or a small neural network, μ is a fixed (potentially task-conditioned) policy, and w_i is the state-independent embedding of task i.

Compression of features DIVRS relies on finding a compressed representation ψ_ω of the state. If the representation is insufficiently compressed, i.e., contains too many random projections from irrelevant features of the environment; or if the representation is insufficiently general, i.e., captures spurious but controllable features of the environment (such as changes in the background relative to the agent), the skills that learn to control the resulting features may not be useful. Hence regularization and compression play a key role in making this approach work.

Information bottleneck methods (Tishby et al., 2000) encourage learning representations that are both general and do not contain projections from irrelevant parts of the state space. These methods capture the notion of a minimal sufficient statistic by maximizing the information conserved about the relevant prediction (in our case the value function),

while maximizing the amount of information lost between the raw state and the encoded state: $\min_\omega I(s; \psi_\omega(s) - I(\psi_\omega(s); V_i^\pi(s)))$. This can be done explicitly (Alemi et al., 2016; Igl et al., 2019a), though there is some evidence that it occurs naturally in capacity constrained deep networks (Tishby & Zaslavsky, 2015; Shwartz-Ziv & Tishby, 2017), which we found give sufficiently terse encodings. To promote this further, we make use of regularisation during training (Cobbe et al., 2019) and, for the second phase, take the compressed, ψ_ω from a value function that is distilled after learning (Rusu et al., 2015), in order to capture features corresponding to a minimal sufficient statistic for value.

The pseudo-code for this part of the method, can be found in Algorithm 3. ψ_ω is trained through a regression objective to contain minimal information to predict the outputs of $\psi_{\hat{\omega}}$, despite being a lower dimension embedding. This is accomplished by training a reconstruction network d_z that takes the compressed representation ψ_ω and projects it into the original representation space.

3.2 Phase Two

Learning skills In the second phase, without utilizing rewards, we train a set of skills using VOD methods. Instead of providing the discriminator directly with prior knowledge about the space in which the skills should be diverse such as an x-y prior—which may not be accessible in settings environments where the skills need to be learned—we instead employ the compressed features $\tilde{\psi}_\omega$ learned in the first phase. While our approach is compatible with any VOD method, in our experiments we employ either forward or reverse MI forms of DIAYN (Eysenbach et al., 2019; Campos et al., 2020). The discriminator and skills are trained in parallel, with the discriminator trained to minimize the prediction loss at each state independently–i.e., $\mathcal{L}_\tau(\phi) = \sum_{t \in \{1..T\}} \mathcal{L}_t(\phi)$, where $\mathcal{L}_t(\phi) = -\log q_\phi(t)$. The discriminator loss at a timestep t is $q_\phi(t) = q_\phi(\psi_\omega(s_t)|z_t)$ for the forward MI objective, and $q_\phi(t) = q_\phi(z_t|\psi_\omega(s_t))$ for reverse MI. The resulting pseudo-reward for the skill is $r_t = \log q_\phi(t) - \log Z_t$ with normalisation term $Z_t = \sum_z q_\phi(\psi_\omega(s_t)|z)P(z)$ for the forward objective and $Z_t = P(z_t)$, with uniform skill prior distribution $P(z)$. The parameters of the state embedding $\psi(s)$ are fixed for the duration of this phase. For the description of forward form objective, see Appendix A.

Skill use DIVRS results in a set of skill policies $\pi_\theta(a|s, z)$ parametrized through z. These policies are learned to visit distinct regions of the learned state space, learning to control exactly the features relevant to the value of task distribution. This means that these skills can be used on unseen downstream tasks $\mathcal{M}$ with the shared structure, either by selecting and fine-tuning the best performing skills or by training a meta-controller $\pi(z|s)$ as in Eysenbach et al. (2019).

3.3 Properties of Learned Skills

The above procedure results in skills that manipulate the environment in accordance with the dynamics of the value function, rather than the state space itself. Indeed, under the assumptions of effective compression and skill learning, as well as the presence of controllable features in the state space that are not informative of state values, skills learned via DIVRS have higher MI with state values than skills learned directly from the observation space. We express this formally in the following theorem.

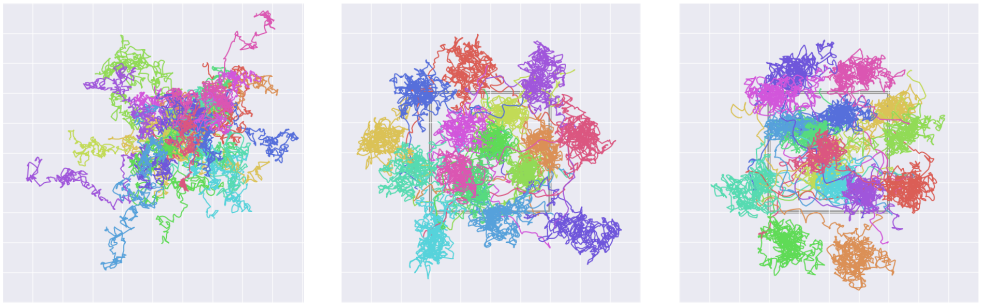
Theorem 3.1. *For a fixed distribution over states and skill priors, let $\{U, \zeta\}$ be a partition of state features, such that U represents features useful to value function prediction, with MI $I(U, V) = H(V)$, while features in ζ are sampled independently of those in U, with $I(U, \zeta) = 0$. After learning sets of the same number of skills using both DIAYN and DIVRS on the fixed distribution, let Z_S represent the random variable corresponding with the DIAYN skill index active at a given time step. Let Z_ψ represent the active skill index of skills learned using DIVRS. Assuming that perfect discriminability is achieved: $H(Z_\psi|\psi(S)) = H(Z_S|S) = 0$, that perfect compression is achieved: $H(V) = I(V; S) = I(\psi(S); S) = I(\psi(S); V)$, and that DIAYN skills learn to manipulate some features in ζ: $I(\zeta; Z_S) \geq 0$, we have:*

$$I(V; Z_\psi) > I(V; Z_S).$$

The proof can be found in Appendix B. Intuitively, the theorem holds because DIAYN learns skills to control elements of state space that may be uncorrelated with state values, while DIVRS constrains skills to only manipulate features relevant to the value function. Given that the methods have the same number of skills, this means that all of the information in Z_ψ pertains to V, while some of the information in Z_S does not.

4 Empirical Evaluation

In this section, we examine the empirical results of our proposed method. Our aim is to constrain the space in which the skills are diverse and thus obtain qualitatively and quantitatively better skills. In addition to the MuJoCo results



(a) Reverse MI (b) Forward MI with $x-y$-prior (c) Forward MI with DIVRS

Figure 1: Comparison of trajectory traces in x-y plane of 16 different Ant skills: (a) reverse MI, (b) forward MI with x-y prior, (c) forward MI with DIVRS. The grey square is for scale, denoting a 2m$\times$2m area. Lines with the same color denote different rollouts of the same skill. Hand-crafting features (b) or transfer (c) were necessary to obtain skills that are consistent in and distinguished by visitation of different areas of $x-y$ plane.

presented here, we also ran the method on simple gridworld environments. These results can be found in Appendix D. The code needed to reproduce all experiments here will be released with the final version of the paper.

4.1 MuJoCo Experiments

We start by examining skills trained on MuJoCo Ant (Todorov et al., 2012; Brockman et al., 2016), which has previously required hard-coded restrictions on the observation space of the discriminator (an x-y prior) to obtain skills useful for complex navigation tasks (Eysenbach et al., 2019; Sharma et al., 2020). In this way, the Ant environment is a good example of a failure mode for DIAYN that DIVRS is perfectly suited to solve. As source tasks, we use three simple navigational tasks. In each, the agent needs to reach a specific goal g, chosen from three fixed goal positions. The agent receives a sparse reward and the episode terminates when the agent is sufficiently close to the goal. We found that in MuJoCo tasks we tried, quality of skills obtained using the value features learned by policy evaluation of a random policy was as good as quality of skills obtained from value features of optimal policies, hence the results reported here use the latter. The value of a state in these tasks is determined mostly by the distance of the agent's center of mass to the goal; hence a representation that contains information needed for all three value functions needs to capture information about the position of the agent.

If the first phase of training, we use PPO (Schulman et al., 2017) to learn a goal-conditional value function $V(s,g)$. The value function is parametrized as $V(s,g) = f_v(\psi(s) \odot w(g))$, where the function $f_v(\cdot)$ receives an element-wise multiplication of the state and goal embeddings ($\psi(s)$ and $w(g)$ respectively). To ensure sufficient generality and feature compression , we distill the embedding $\psi(s)$, before passing it to the discriminator in the second phase. Distillation is performed on data gathered from a random policy. A network with fewer parameters is trained to predict the output of ψ through a narrow bottleneck layer $\bar{\psi}$. This bottleneck layer is then used as state input to the discriminator in the following phase.

In the second phase of training, we use SAC Haarnoja et al. (2018) with the the forward MI objective to learn skills. We chose forward MI in order to ensure skills behaved consistently across the state space (see Appendix C for a more detailed discussion) and SAC as it was the algorithm of choice for IT skill discovery in related work (Eysenbach et al., 2019; Sharma et al., 2020). In forward MI experiments, we model the skill-conditional state distribution $q_\phi(s|z)$ with a multivariate Gaussian in the learned space (or predefined space when using $x-y$ prior) with fixed uniform diagonal covariance and learned location parameter. The scale of variance was tuned as a hyperparameter, though it could in principle be optimized via the discriminator objective alongside the mean. More details about the experiment design can be found in Appendix F.

4.1.1 Qualitative Skill Evaluation

To analyze learned skills, we look at the x-y traces and saliency maps (Simonyan & Zisserman, 2014) of the trained discriminator. These x-y traces show how well the skills cover the x-y plane and if they consistently reach certain areas. Saliency maps of the discriminator 2 show which features of the state are used to discriminate skills. Figure 1 compares x-y traces of vanilla DIAYN skills, skills of DIAYN with an x-y prior, and skills obtained with DIVRS. As reported by Eysenbach et al. (2019); Sharma et al. (2020), baseline skills trained with DIAYN do not venture far from the initial state position, whereas both DIAYN with an x-y prior and DIVRS explore the x-y plane well, though

DIVRS is able to do so using learned, rather than explicit encoding of these relevant features. Note that we do not show skills trained on Forward MI without the x-y prior, as we found the high dimensionality of state space in this case to be too prohibitive for learning diverse skills. Figure 2 shows that vanilla DIAYN skills learn to be primarily differentiated via easy-to-control features like joint angles. Providing the discriminator a representation pre-trained on the source tasks limits the discriminator to relying on agent position. For further insight into the qualitative nature of the skills learned by DIVRS, we performed a comparison across VOD methods in Appendix A, and investigate the importance of the distillation phase and the degree of compression achieved in Appendix E.

4.1.2 Sparse Downstream Task with HRL

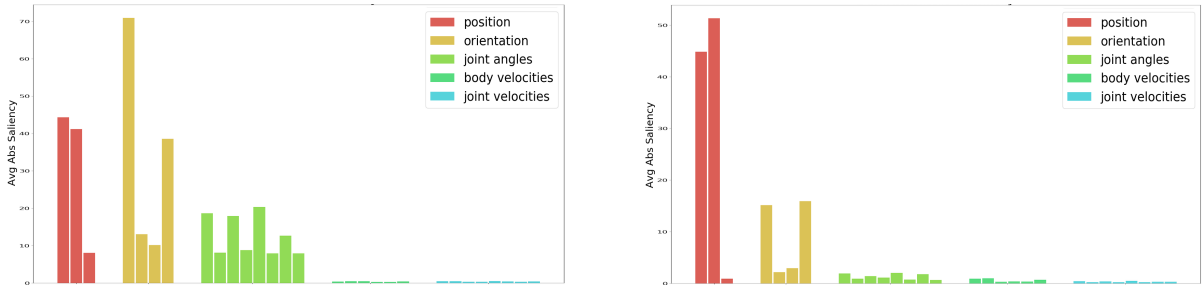
Next, we evaluate the effectiveness of skills learned by DIVRS on a challenging task that is difficult to solve without skills. In particular, we apply them to the sequential sparse navigation task from (Eysenbach et al., 2019). This task is challenging to solve even with skills if the space in which exploration is needed has not been specified. The agent starts at $(0, 0)$ and receives a sparse reward for reaching the corners of an axis-aligned square of length four. These points must be reached in a fixed order held constant across all episodes.

We employ a variant of SAC designed to work with discrete action spaces (Christodoulou, 2019) in the Semi-Markov Decision Process (SMDP) (Puterman, 2014) induced by our skills over the task MDP. Skills, which were executed for 1000 steps during training, are now terminated after (fewer) fixed time steps, before control is returned to our hierarchical agent. This skill duration was tuned as a hyperparameter across independent rollouts of the HRL learning algorithm. In addition to the state space provided to the skills, in order to ensure that states remain Markov, the skill coordinator policy receives a one-hot representation of the number of goals reached. We train both the flat and hierarchical policies for five million primitive time steps, although hierarchical methods trained at the SMDP level can only make use of data gathered at the hierarchical decision points.

Figure 3b shows the results of this experiment. The skills learned by DIVRS consistently achieve the maximal reward. This is comparable and even surpasses performance on the task with skills learned through explicitly specifying the space in which exploration is useful, though we have only specified a binary reward signal. We see that skills learned without any specification are generally unsuccessful in the task, as they do not explore the relevant feature space efficiently. Similarly, directly training flat models is unsuccessful in achieving consistent returns. For more details on the hyperparameters used in this experiment, see Appendix H.

4.1.3 Few-Shot Task with Skill Selection

In this section, we evaluate the zero-shot performance of learned skills on a sparse reward goal-seeking locomotion task. Goals are sampled from within the 4×4 box centred at the origin, and remain fixed for the evaluation period. We report performance averaged over ten different goals sampled from this distribution. A reward of 1 is given when the agent is within $1m$ of the goal and the episode is terminated. Rewards are zero otherwise. The DIVRS skills are compared to DIAYN and x-y prior DIAYN skills for both forward and reverse modes, as well as a randomly initialized policy. Like Eysenbach et al. (2019), we evaluate all skills and choose the best one as a solution to the task. To account for the additional environment steps needed to evaluate all 16 trained skills, we fine tune the random policy on each task using SAC with the corresponding number of environmental interactions.

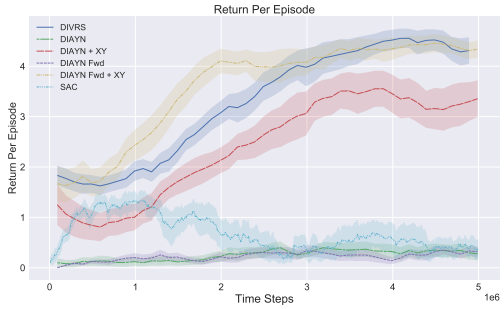


(a) DIAYN (b) DIVRS

Figure 2: Analysis of features used to differentiate skills by visualizing discriminator saliency. The features are grouped into positions, orientation, joint angles, body velocities, and joint velocities. Position features are much more salient for prediction with DIVRS (b) than DIAYN (a).

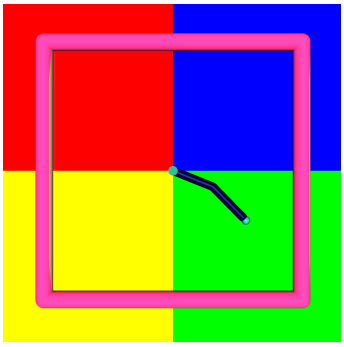
Method	Avg Return
DIVRS	0.954 ± 0.031
DIAYN	0.360 ± 0.084
DIAYN + x-y-prior	0.972 ± 0.012
Forward MI DIAYN	0.384 ± 0.169
Forward MI DIAYN + x-y-prior	0.947 ± 0.040
SAC	0.174 ± 0.051

(a) Zero-shot performance on unseen goals.

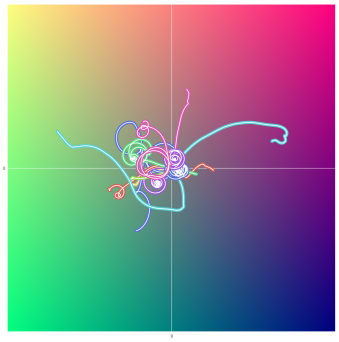


(b) Performance on sparse sequential task

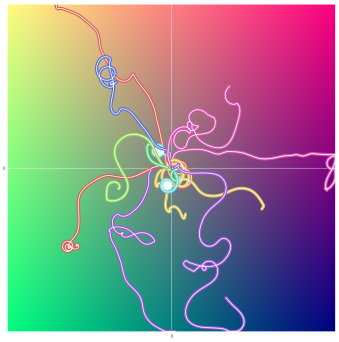
Figure 3: Results of downstream learning experiments: (a) zero-shot performance on unseen goals and (b) learning curves on sparse sequential task. Skills learned with DIVRS are significantly more effective on both tasks compared to both the flat baseline and skills learned in the original state space, while performing as well or better than methods utilizing hand-crafted features.



(a) Pixel Reacher Environment



(b) Skills Learned with DIAYN



(c) Skills Learned with DIVRS

Figure 4: Results on Pixel Reacher environment. The environment is illustrated in (a): the color of Reacher's tip and base is modified towards one of the four colors, with the colour determined by quadrant currently visited by the tip. The trajectories in the colour space of eight skills obtained with DIAYN and DIVRS are shown in (b) and (c) respectively. Lines with the same colour denote different rollouts of the same skill. DIVRS improves both coverage and consistency of skills in the subspace relevant for performance. Best viewed in colour.

The results are shown in Table 3a. As DIAYN skills do not cover much of the x-y space, they infrequently end up close to the goal. The random policy is able to reach the new goals occasionally, but the sparse learning problem is challenging, and thus the SAC agent is not able to learn with the little data available. By contrast, by covering the x-y space well, skill sets learned with DIVRS—and those learned using DIAYN with the x-y prior—are able to reach states in proximity to most sampled goals, leading to consistent rewards.

4.2 Pixel MuJoCo Experiments

In the tasks considered above, using hand-engineered features is feasible, as the relevant features are trivially selected from the available state features. To evaluate DIVRS in a more challenging setting, we consider a task where the relationship between state and relevant features is more complex because the observation space consists of images. We run our experiments on a variant of the MuJoCo Reacher with pixel observations, loosely inspired by the cooking robot example in Section 1.

The environment In addition to the original Reacher state, the agent must learn to manipulate an additional state subspace, given by the plane $[0, 1]^2$. We can visualise this additional subspace as a colour plane in RGB colour space, with a fixed blue chromaticity, and green and red chromaticities varying proportionally to the position in the plane. See the background of the trace plots in Figure 4 (b, c) for a rendering of this space. We colour the tip and the base of the Reacher agent according to its current position on this plane, so that the position in colour space is observable in the agent's (pixel) state space. We can imagine this colour space as an abstraction of flavour space for the cooking robot.

Depending on the position of the Reacher tip in the original state space, the colour of the tip and base change accordingly (this heavily abstracts the cooking process of moving around the kitchen, picking up spices, etc). As the Reacher tip moves further from the origin, the flavour changes more quickly, in accordance with the Euclidean norm of the tip position. To make the environment even more challenging, the direction of movement in colour space corresponds to the position of the tip (which falls in $[-1, 1]^2$), randomly rotated by a multiple of $90°$ each episode. This rotation is represented in pixel space with four coloured panels below the agent, one in each quadrant, where each panel corresponds to movement in that direction in colour space (see Figure 4 (a)).

A visualisation of the environment is shown in Figure 4a. As in Hafner et al. (2019), the observations have been down-scaled to a 64×64 resolution, though here we use three-channel colour pixels. We concatenate two consecutive states to capture the joint velocities of the agent.

Learning Skills As before, the source tasks involve navigation to one of three fixed goals. However, now the goals exist in colour space, corresponding to a few sample dishes that the agent can be trained to learn. The reward on each of these source tasks is dense, corresponding to the negative distance from the agent's colour state to the goal. We use the same learning algorithm and the value function parametrization as in the MuJoCo Ant experiments; the main difference is that now only eight skills are learned, and the pixel observations are embedded with a convolutional neural network. More details on the network architecture and experiment design can be found in the Appendix G

To analyze learned skills, we examine the x-y traces in colour space, across two rollouts of each skill. The results are shown in Figure 4. For each method, we select the set of skills with the highest mutual information between skills and (embedded) states as measured on test rollouts. Since the observation space is rich, skills obtained through naive IT objectives can learn to be differentiated through many features, including body positions and velocities. In fact, in colour space, the DIAYN agent has reduced consistency of skills, and reduced state coverage compared to a random agent. By contrast, skills learned with DIVRS are much more consistent and are primarily differentiated by the agent's position in colour space.

5 Related Work

Unsupervised Skill Discovery Several methods have been developed for the automatic discovery of skills in RL. Particularly relevant, due to their unsupervised nature, is a category of information-theoretically motivated, reward-agnostic methods, which optimise policies according to various criteria, centered around notions of discriminability (Gregor et al., 2016; Eysenbach et al., 2019; Achiam et al., 2018; Sharma et al., 2020). These techniques are generally highly underconstrained, in that many sets of skills satisfy the optimization objective, meaning that they produce uninteresting or useless skills in practice, unless significant task-specific guidance is provided (Achiam et al., 2018). Hausman et al. (2018) combine discriminability and task rewards; however, they aim to find multiple solutions to the specified task.

Transfer Learning Several prior works also learn skills from a set of tasks (Thrun & Schwartz, 1995; Pickett & Barto, 2002). Frans et al. (2018) extract skills from a set of tasks, but unlike our approach, are restricted to downstream tasks that can be solved by a composition of the source task policies. Igl et al. (2019b) relax this restriction by also learning termination functions, allowing the composition of sub-policies from the source tasks. More widely, KL-regularization can be used to transfer knowledge between tasks Teh et al. (2017); Galashov et al. (2018); Tirumala et al. (2019), by using skill hierarchies to implement various inductive biases. Without the complications that can arise by learning temporal abstractions, Wulfmeier et al. (2019) show that sharing policies across closely related tasks can speed up training. However, all of these approaches are restricted to policies that are the same or close (as measured by a KL-divergence) to those on the source tasks. By contrast, we learn policies that are as diverse as possible, while using source tasks only to identify the relevant part of the state space.

Other Similar settings with factored reward functions $r_w(s, a) = \phi(s, a)^T w$ have been studied (Barreto et al., 2017; Borsa et al., 2019), although, there is less emphasis on learning a compressed state representation as it does not benefit from lower dimensionality of $\phi(s)$. In addition, it is typically required that the weights w of downstream tasks are learned or known in advance. The assumption that under certain smoothness conditions, optimal value functions can be factored as $V^*(s, g) = \psi(s)^T w(g)$ is also employed by Schaul et al. (2015). For discussion on related work in HRL and learning state abstractions, see Appendix I.

6 Discussion

One of the limitations of our approach is that the user has to select a set of source tasks, or when available, randomly sample from the distribution of tasks of interest. Selection of source tasks is already a common practice in transfer learning literature, however, in some cases it may not be clear what the related source tasks are or the reward in source tasks will not be encountered sufficiently often to learn value-relevant features. Additionally, our two phase approach is not well suited to settings where the task distribution changes over time. This could be approached in future work by iterating between phases of skill learning and skill usage, or by building hierarchies of skills, again alternating between skill construction and data collection. When the relevant features for the diversity objective are known and can be easily specified, DIVRS is not an appropriate method to use, as the additional burden of learning good features is unnecessary. Furthermore, tuning of hyperparameters can be a challenge: parameters in later learning phases can be tuned independently, but hyperparameters in early phases need to be tuned in tandem with downstream hyperparameters, adding to optimisational complexity. Our method also relies on good exploration in both phases of the algorithm to ensure good state coverage. This is an important but mostly an orthogonal problem to the one our method is solving. In our experiments, using a random policy was sufficient to ensure good state coverage, but more generally, methods such as marginal state matching (Lee et al., 2019) or pseudo-counts Bellemare et al. (2016) can be used to ensure enough exploration.

7 Conclusion

In this work, we developed a representation-learning approach for the automatic construction of sets of skills that enable agents to efficiently traverse the space of features that reflects performance on a set of related tasks. In doing so, we also overcame a core limitation of modern diversity-based skill discovery methods: under-specification of the space in which diversity is meaningful. In complex domains, such an objective can often trivially be satisfied by skills that offer little utility on downstream tasks. We addressed this problem by using pretraining to learn a good state representation for the discriminator, which focuses more on features that are relevant for the tasks of interest. Our empirical results demonstrate that, when compared to flat baselines and skill learned without pretraining, these representations lead to skills that provide better coverage of the relevant state space and lead to high performance on challenging sparse reward downstream tasks.

Acknowledgments

Kristian Hartikainen is funded by the EPSRC.

References

David Abel, David Hershkowitz, and Michael Littman. Near optimal behavior via approximate state abstraction. In *International Conference on Machine Learning*, pp. 2915–2923. PMLR, 2016.

David Abel, Dilip Arumugam, Kavosh Asadi, Yuu Jinnai, Michael L Littman, and Lawson LS Wong. State abstraction as compression in apprenticeship learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3134–3142, 2019.

David Abel, Nate Umbanhowar, Khimya Khetarpal, Dilip Arumugam, Doina Precup, and Michael Littman. Value preserving state-action abstractions. In *International Conference on Artificial Intelligence and Statistics*, pp. 1639–1650. PMLR, 2020.

Joshua Achiam, Harrison Edwards, Dario Amodei, and Pieter Abbeel. Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299*, 2018.

Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*, 2016.

Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 166–175. JMLR. org, 2017.

André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. In *Advances in neural information processing systems*, pp. 4055–4065, 2017.

Marc G Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. *arXiv preprint arXiv:1606.01868*, 2016.

Diana Borsa, Andre Barreto, John Quan, Daniel J. Mankowitz, Hado van Hasselt, Remi Munos, David Silver, and Tom Schaul. Universal successor features approximators. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1VWjiRcKX.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

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

Víctor Campos, Alexander Trott, Caiming Xiong, Richard Socher, Xavier Giro-i Nieto, and Jordi Torres. Explore, discover and learn: Unsupervised discovery of state-covering skills. *arXiv preprint arXiv:2002.03647*, 2020.

Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for openai gym. https://github.com/maximecb/gym-minigrid, 2018.

Petros Christodoulou. Soft actor-critic for discrete action settings, 2019.

Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. *arXiv preprint arXiv:1912.01588*, 2019.

Thomas G Dietterich. The maxq method for hierarchical reinforcement learning. In *ICML*, volume 98, pp. 118–126. Citeseer, 1998.

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

Kevin Frans, Jonathan Ho, Xi Chen, Pieter Abbeel, and John Schulman. META LEARNING SHARED HIERARCHIES. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SyX0IeWAW.

Alexandre Galashov, Siddhant Jayakumar, Leonard Hasenclever, Dhruva Tirumala, Jonathan Schwarz, Guillaume Desjardins, Wojtek M Czarnecki, Yee Whye Teh, Razvan Pascanu, and Nicolas Heess. Information asymmetry in kl-regularized rl. 2018.

Sandeep Goel and Manfred Huber. Subgoal discovery for hierarchical reinforcement learning using learned policies. In *FLAIRS conference*, pp. 346–350, 2003.

Anirudh Goyal, Riashat Islam, Daniel Strouse, Zafarali Ahmed, Matthew Botvinick, Hugo Larochelle, Yoshua Bengio, and Sergey Levine. Infobot: Transfer and exploration via the information bottleneck. *arXiv preprint arXiv:1901.10902*, 2019.

Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pp. 2555–2565. PMLR, 2019.

Anna Harutyunyan, Will Dabney, Diana Borsa, Nicolas Heess, Remi Munos, and Doina Precup. The termination critic. *arXiv preprint arXiv:1902.09996*, 2019.

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

Maximilian Igl, Kamil Ciosek, Yingzhen Li, Sebastian Tschiatschek, Cheng Zhang, Sam Devlin, and Katja Hofmann. Generalization in reinforcement learning with selective noise injection and information bottleneck. In *Advances in neural information processing systems*, pp. 13978–13990, 2019a.

Maximilian Igl, Andrew Gambardella, Nantas Nardelli, N Siddharth, Wendelin Böhmer, and Shimon Whiteson. Multitask soft option learning. *arXiv preprint arXiv:1904.01033*, 2019b.

Nicholas K Jong and Peter Stone. State abstraction discovery from irrelevant state variables. In *IJCAI*, volume 8, pp. 752–757. Citeseer, 2005.

Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems*, pp. 3675–3683, 2016.

Sascha Lange and Martin Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In *The 2010 international joint conference on neural networks (IJCNN)*, pp. 1–8. IEEE, 2010.

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

Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for mdps. *ISAIM*, 4: 5, 2006.

Evan Zheran Liu, Aditi Raghunathan, Percy Liang, and Chelsea Finn. Decoupling exploration and exploitation for meta-reinforcement learning without sacrifices. *arXiv preprint arXiv:2008.02790*, 2020.

Amy McGovern and Andrew G. Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pp. 361–368, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-778-1. URL http://dl.acm.org/citation.cfm?id=645530.655681.

Ofir Nachum, Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. *arXiv preprint arXiv:1805.08296*, 2018a.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Near-optimal representation learning for hierarchical reinforcement learning. *arXiv preprint arXiv:1810.01257*, 2018b.

Ofir Nachum, Haoran Tang, Xingyu Lu, Shixiang Gu, Honglak Lee, and Sergey Levine. Why does hierarchy (sometimes) work so well in reinforcement learning? *arXiv preprint arXiv:1909.10618*, 2019.

Junhyuk Oh, Satinder Singh, and Honglak Lee. Value prediction network. *arXiv preprint arXiv:1707.03497*, 2017.

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

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International conference on machine learning*, pp. 1312–1320, 2015.

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

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

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised skill discovery. *ICLR*, 2020.

Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information, 2017.

David Silver, Hado Hasselt, Matteo Hessel, Tom Schaul, Arthur Guez, Tim Harley, Gabriel Dulac-Arnold, David Reichert, Neil Rabinowitz, Andre Barreto, et al. The predictron: End-to-end learning and planning. In *International Conference on Machine Learning*, pp. 3191–3199. PMLR, 2017.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CVPR*, 2014.

Özgür Şimşek and Andrew G Barto. Skill characterization based on betweenness. In *Advances in neural information processing systems*, pp. 1497–1504, 2009.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 4499–4509, 2017.

Chen Tessler, Shahar Givony, Tom Zahavy, Daniel J Mankowitz, and Shie Mannor. A deep hierarchical approach to lifelong learning in minecraft. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Sebastian Thrun and Anton Schwartz. Finding structure in reinforcement learning. In *Advances in neural information processing systems*, pp. 385–392, 1995.

Dhruva Tirumala, Hyeonwoo Noh, Alexandre Galashov, Leonard Hasenclever, Arun Ahuja, Greg Wayne, Razvan Pascanu, Yee Whye Teh, and Nicolas Heess. Exploiting hierarchy for learning and transfer in kl-regularized rl. *arXiv preprint arXiv:1903.07438*, 2019.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle, 2015.

Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

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

Markus Wulfmeier, Abbas Abdolmaleki, Roland Hafner, Jost Tobias Springenberg, Michael Neunert, Tim Hertweck, Thomas Lampe, Noah Siegel, Nicolas Heess, and Martin Riedmiller. Regularized hierarchical policies for compositional transfer in robotics. *arXiv preprint arXiv:1906.11228*, 2019.

Appendices

A Forward Mutual Information Objective

In our experiments on continuous control tasks using DIVRS, we employ the *forward* mutual information objective, which derives from the decomposition of mutual information between states and skill indicies as:

$$I(S;Z) = H(S) - H(S|Z)$$

Since directly optimizing for mutual information is intractable, we instead optimize the approximate lower bound:

$$\begin{aligned} I(S;Z) &= \mathbb{E}_{s,z}[\log p(s|z)] - \mathbb{E}_s[\log p(s)] && (5) \\ &\geq \mathbb{E}_{s,z\sim p(z,s)}[\log q_\phi(s|z)] - \mathbb{E}_{s\sim p(s)}[\log p(s)] && (6) \end{aligned}$$

Where the discriminator network $q_\phi(s|z)$ is trained to approximate the probabilities in the conditional entropy term, predicting states from skill indices. Note that this form does not permit a true variational bound due to the fact that the state prior, $P(S)$ as well as $P(S|Z)$ must be approximated with the stationary distribution.

B Proof of Theorem 1

Here, we demonstrate that, under a fixed distribution over states and fixed skill priors, with assumptions that ensure good compression and good skill learning, and the presence of controllable, noisy features in the state, the skills learned by DIVRS will be more informative of the value function than skills learned without pretraining.

In an MDP, for a fixed state distribution, we can consider the value function to be a transformation of the random variable S, corresponding to the state. This gives us a distribution over values of states, $P(V(S))$. In reference to the information bottleneck model of data generation, we invert this relationship, assuming that a randomly sampled label (in our case the value function) generates a distribution over states. Our goal in pretraining, is to then capture a minimally sufficient statistic of V from S. To do so we optimize the information bottleneck objective (Tishby et al., 2000) for a parameterized embedding of state, $\psi_\theta(s)$:

$$\min_\theta I(\psi_\theta; S) - \beta I(\psi_\theta; V)$$

Here we explicitly partition the state space into components that can be used to predict the value U, and those that are sampled independently of U, given by ζ (i.e. $I(\zeta, U) = 0$). While this is not always possible, it captures the notion that some components of state are not very informative of the value function (imagine joint angles on a MuJoCo task in which the agent must reach a specific point in space). S in it's entirety is then encoded into the latent representation, ψ, which represents a minimal sufficient statistic of V from features in S. In order to express that this encoding is done perfectly, we introduce the following assumption:

Assumption B.1. ψ contains exactly the information from S needed to recreate V, and no other information, that is:

$$H(V) = I(V; S) = I(\psi(S); S) = I(\psi(S); V).$$

Assumption B.1 implies that both S and $\psi(S)$ can perfectly predict V, and that the only information in both S and $\psi(S)$ is the information about V. Given that ψ is a (deterministic) embedding of S used to predict V, which is itself a deterministic function of S, this assumption is satisfied, so long as ψ does not contain spurious projections from S, and so long as ψ is rich enough to capture the complexity of $V(S)$. In our running example, the locomotion task, this assumption corresponds to the idea that $\phi(S)$ encodes the position of the agent, which is sufficient information to predict the value of S. Further, it suggests that spurious features, such as joint angles are not included in $\phi(S)$. This assumption will be satisfied based on experimental design choices, such as the dimension of $\phi(S)$, and the reward function in the source tasks.

We build skills from either the raw state space (as in DIAYN), Z_S, or from our compressed representation, leading to skills Z_ψ, where these random variables correspond to the skill index active at the current state. Skill indices are then distributed across states (or embeddings) to maximize the corresponding mutual information ($I(S; Z_S)$ or $I(\psi(S); Z_\psi)$) for a fixed (uniform) prior distribution over skill indices and our fixed distribution over states. For the purposes of the proof we assume that this process occurs perfectly–the skill index can be predicted perfectly from the state or embedding that it derives from:

Assumption B.2. Skill indicies are perfectly predictable from their corresponding state or embedding variable:

$$H(Z_S|S) = H(Z_\psi|\psi(S)) = 0.$$

In practice, this assumption is almost never going to be perfectly realised, as the set of initial states will always have ambiguous skill labels, since all skills can be initialised in them. However, once skills are differentiated, this will be nearly satisfied, as the agent will immediately move away from ambiguous states towards skill-specific states.

From assumption B.2, it follows that $I(Z_S; S) = I(Z_\psi; \psi(S))$, since $H(Z_S) = H(Z_\psi)$, given that the skill indicies have the same marginal distribution by construction.

The entire process can be visualised via the graphical model given by Fig. 5 at every time step.

Our final assumption is concerned with the fitting of the skills to the value-irrelevant features, ζ. We assume that the skill policies $\pi(s|z_s)$ are able to control features in ζ, and thus the skills Z_S learn to contain at least some information about ζ when maximizing the mutual information between states and skill indices:

Assumption B.3. Skills to manipulate features in ζ:

$$I(Z_S; \zeta) > 0.$$

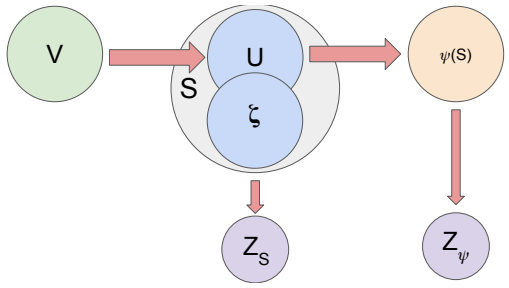


Figure 5: Graphical model for each time step while learning skills from both pretrained representations and raw states. $P(S)$, $P(Z_S)$, and $P(Z_\psi)$ are all fixed.

This assumption can usually be expected hold true. The only environments in which it will not is when all spurious features that are not indicative of task value also cannot be controlled by the agent. We can imagine environments with "TV static" (Burda et al., 2018) to thus not satisfy this assumption. In our control example, however, the agent joint angles satisfy this assumption because they can be controlled.

For simplicity's sake, in our description, we have considered only the value function of a single task, though the proof that follows easily extends to a multi-task setting, with value $V' = \mathbb{E}[V]$, where the expectation is over the task distribution. The only requirement is that the features in ψ are sufficient to predict the expected value across tasks.

With our assumptions collected, we can move on to the proof at hand.

Theorem B.4. *For a fixed distribution over states and skill priors, let $\{U, \zeta\}$ be a partition of state features, such that U represents features useful to value function prediction, with MI $I(U, V) = H(V)$, while features in ζ are sampled independently of those in U, with $I(U, \zeta) = 0$. After learning sets of the same number of skills using both DIAYN and DIVRS on the fixed distribution, let Z_S represent the random variable corresponding with the DIAYN skill index active at a given time step. Let Z_ψ represent the active skill index of skills learned using DIVRS. Given assumptions B.1, B.2, and B.3, we have:*

$$I(V; Z_\psi) > I(V; Z_S).$$

Proof. We start by applying Bayes' rule to $H(Z_\psi|V)$:

$$\begin{aligned}
H(S_\psi|V) &= H(Z_\psi|V, \psi(S)) + H(\psi(S)|V) - H(\psi(S)|Z_\psi, V), \\
&= H(\psi(S)|V) - H(\psi(S)|Z_\psi, V), && \text{(Assumption B.2)} \\
&\leq H(\psi(S)|V) - H(\psi(S)|Z_\psi, V, S), \\
&\leq H(\psi(S)|S) - H(\psi(S)|S). && \text{(Assumption B.1 and } \psi \text{ is a deterministic function of } S) \\
&\leq 0.
\end{aligned}$$

However, since entropy is non-negative, it must be that $H(z_j|V) = 0$. From this it follows that:

$$\begin{aligned}
I(Z_\psi; V) &= I(Z_\psi; \psi(S)) = I(Z_S; S), && \text{(Assumption B.2)} \\
&= H(Z_S) - H(Z_S|U, \zeta), \\
&= H(Z_S) - H(\zeta|U, Z_S) - H(Z_S|U) + H(\zeta|U), \\
&= I(Z_S; U) + I(Z_S; \zeta|U), \\
&\geq I(Z_S; V) + I(Z_S; \zeta|U). && \text{(Data Processing Inequality and independence of } U \text{ and } \zeta)
\end{aligned}$$

By assumptions B.2 and B.3, we have:

$$I(Z_S; \zeta|U) = I(Z_S; \zeta) - I(U; \zeta) > 0,$$

which gives us the desired result. □

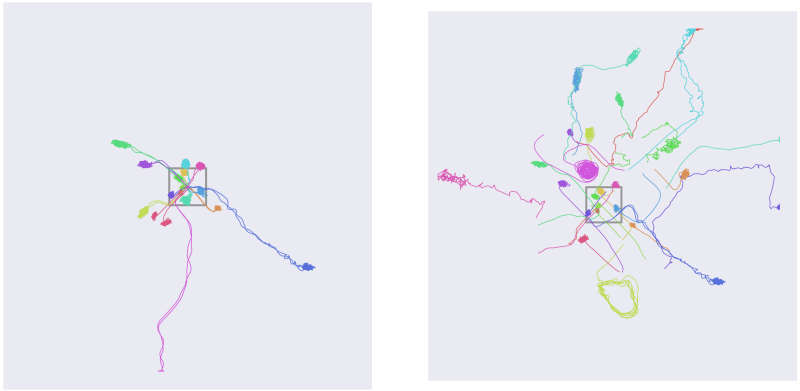
C Forward Mutual Information and Skill Consistency

When training agents from a fixed starting state, we observed that skills learned using reverse MI would suffer during the HRL learning phase when skills were selected by the SMDP-level policy in states which they never observed. These skills would have undefined behaviour, and in the Ant environment, would often cause the agent to "freeze", taking

actions that left the agent largely static. In order to correct for this, two changes were needed: we switched to training skills on random initial states, as well as training skills using the forward mutual information objective.

For illustrative purposes, we have examined this effect in a simple point mass environment. The agent is represented as a point in the x-y plane. It can apply a velocity in $[-1, 1]$ in each dimension independently. Because position is the only feature, applying variational skill discovery methods leads to skills that move to distinct points in the plane.

Figure 6 demonstrates the effect of applying skills learned from a fixed initial state across randomly initialized states in the point mass environment. Skills that appear to be consistent–causing the agent to move to specific states–become ill-defined in states that they were not trained on, leading to erratic behaviour.



(a) Skills learned using reverse MI (fixed initial state)

(b) The same skills, run on a random initial state distribution

Figure 6: Skills learned with a fixed initial state in the point-mass environment. 16 Skills were learned using both reverse mutual information (MI) objectives. Each colour corresponds to a distinct skill. We see that skills learned using reverse MI are consistent when initialized from a fixed state, but each skill behaviour is undefined on states not visited by that skill.

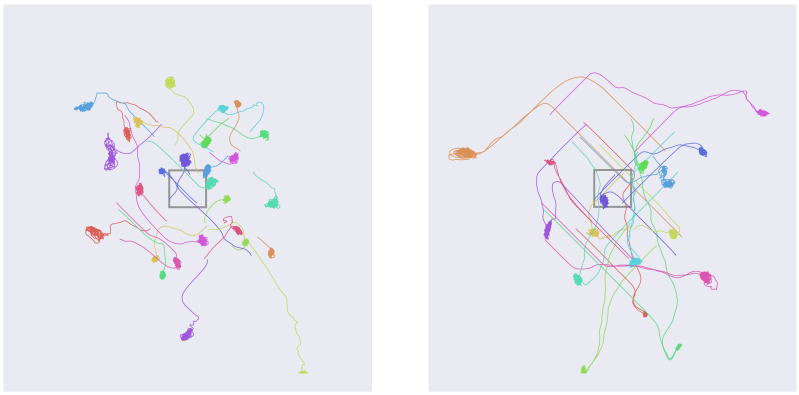
While this effect can be improved by training the agent on a random initial state distribution, use of the reverse mutual information objective still does not lead to skills that behave consistently. The discriminator to classifies states according to the skills they were likely to have been generated by, which can cause an implicit multi-modal partitioning of the state space, where clusters of states that correspond with a given skill are disjoint. In contrast, for the forward mutual information objective, each skill only corresponds to a single distribution over states, which can be parameterized as desired. If a unimodal distribution is chosen, then skill policies will be rewarded for proximity to exactly one state. This leads to consistent skills which reach specific states, regardless of the state that they are initialized from. This means that skills learned with the forward objective are useful for tasks in which skills must be composed, as consistency reduces the overall entropy of the SMDP induced by the skills.

The results of training on a random initial state distribution in the point mass environment using either reverse or forward mutual information objectives are in Figure 7. We find that skills trained using the reverse objective still do not behave consistently, as it is easy for the discriminator to classify several sets of states as belonging to the same skill

While this problem can also be potentially solved by constraining the model class of the discriminator using reverse mutual information, this may be challenging to do in abstract or complex state spaces.

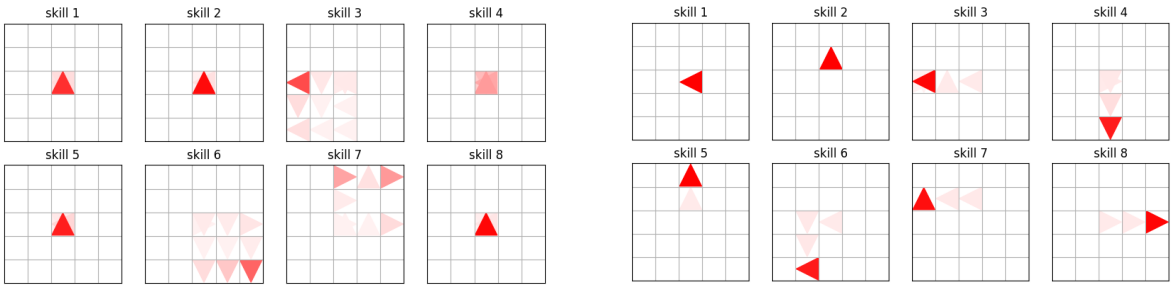
D GRIDWORLD EXPERIMENTS

We start by testing the proposed approach on a modification of a simple 7×7 gridworld from Chevalier-Boisvert et al. (2018) that has been expanded with extra controllable dimensions. In addition to rotating and moving forward, the agent can increase or decrease the value of one of the four added *shadow* dimensions, depending on which of the four cardinal directions it is currently facing. We can interpret them as, e.g., controlling brightness of the wall the agent is currently facing. The purpose of these added dimensions is to demonstrate the effect of expanding the space that the agent can control. Analogous to limb positions in Ant MuJoCo, they are not useful for value functions of navigation tasks, but are easier to control than the agent's position.



(a) Reverse MI (random initial state) (b) Forward MI (random initial state)

Figure 7: Skills learned with randomized initial states in the point-mass environment. 16 Skills were learned using both reverse and forward mutual information (MI) objectives. We see that skills learned using reverse MI can lead to single skills that are disjoint within the state space, when the agent is initialized in a random state. This does not occur when using the forward MI objective. Agents tend to move along diagonals due to a quirk in the action space: because velocity is applied in each dimension independently, moving diagonally this leads to higher overall velocity.



(a) DIAYN (b) DIVRS

Figure 8: Visualizing state visitation in the subspace of interest under 16 different skills trained using DIAYN (Eysenbach et al., 2019) (a) and DIVRS(b). Color intensity of the agent indicates the probability of an agent occupying given direction and position during an episode. Note the differences between the consistency of skills and which features differentiate skills.

To learn which state features are relevant for navigation tasks, we train an entropy-regularized PPO agent (Schulman et al., 2017) on two simple navigation tasks, which involve navigating to one of the two corners of the room. The reward is received only upon task completion. During RL training, we initialize the agent at a random state to ensure generalization of the value function. The grid observation is embedded with convolutional layers, while the non-location observations – direction and added dimensions – are concatenated and embedded with fully connected layers. The two resulting embeddings are concatenated before being passed down to actor and critic heads. The task index identifies which of the two tasks the agent is in and is also embedded with a fully connected network and concatenated with the rest of the embeddings. In these experiments, we found that the embedding distillation step was not needed to achieve a parsimonious representation.

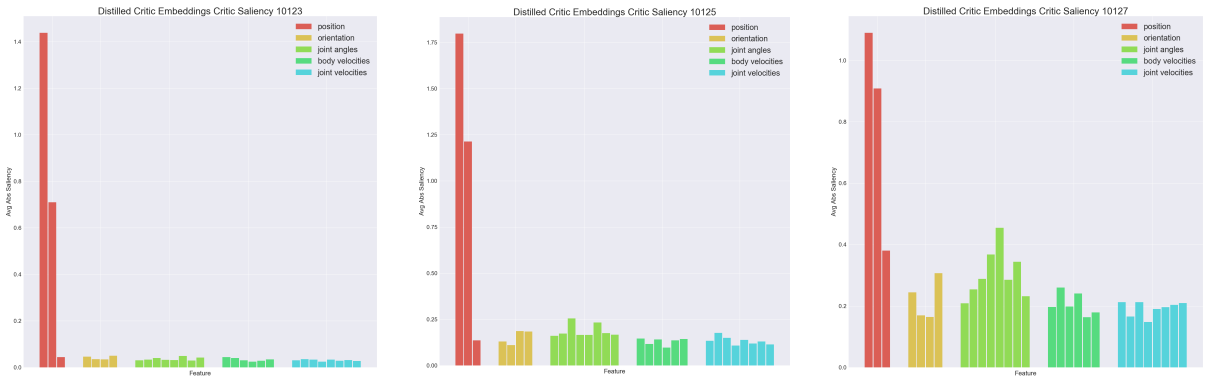
For unsupervised training, we fix the resulting convolutional and fully connected embeddings, and use them as features of the discriminator. For these experiments, we employed the reverse MI form of the DIAYN objective. The discriminator is parametrized as a single hidden layer neural network on top of those features. The skills architecture is similar to the policy in the first phase with the addition of a one-hot encoding of skill identity, which is embedded with fully connected layers and concatenated with the location and non-location embeddings. In experiments with DIAYN, we use the same architecture for the discriminator and skills.

Figure 8 visualizes which states are visited and with what consistency between different skill roll-outs. With the added shadow features, which are easily controlled, most DIAYN skills never leave the initial position (3,3), and instead learn to differentiate by controlling the shadow features or agent orientation. Skills that move in the grid do not reliably

reach a particular position. By pre-training on two navigation tasks, we obtain a representation that ignores added dimensions of the state space as they do not affect the value function. In this case, this leads to differentiation of skills based on the position and orientation of the agent. Fig. 8a shows that five of the eight skills learned with DIAYN do not leave the initial position, and four of them seem to favour the initial orientation as well, instead manipulating the additional shadow dimensions. Those that do move do not do so consistently. By contrast, skills learned with DIVRS move consistently to specific and distinguishable states, both in terms of grid position and orientation.

E Degree of Compression Tests

In order to examine the effect of distillation on compression of the learned embedding, we conduct an experiment where we vary the width of the bottleneck across distilled models. Identical seeding (and thus data, since distillation is performed using a random policy) was used across distillation instances. As a proxy for quality of compression, we examine the saliency maps of the distilled value function. More peaked saliency maps, where few features are highly salient and most are not, correspond to well-compressed embeddings. These saliency maps can be found in fig. 9. We find that as the bottleneck gets smaller, the number of projections from irrelevant features becomes fewer and less significant. This demonstrates the need for the distillation phase, as it further compresses the learned feature space.



(a) Bottleneck size: 4 (b) Bottleneck size: 16 (c) Bottleneck size: 64

Figure 9: Saliency maps of distilled embedding as a function of bottleneck size. We see that a small bottleneck is needed to fully compress the learned representation, removing all projections from irrelevant features.

F MuJoCo Experiment Details

In this section we expand on the details of the experiments conducted in the MuJoCo Ant domain. Following Eysenbach et al. (2019), the gear ratio of the agent is reduced to 30 in order to lead to more stable behaviour. Across all experiments in this section, we employ multi-layer perceptron (MLP) networks with ReLU activation functions for learned functions. We employ the Adam optimizer to perform updates. The observation space includes the x-y position of the agent in order to maintain a Markov state, though we do not privilege these features in any way.

F.1 Pretraining Representations

We learn a representation for a goal seeking agent by initially training on three fixed goals. Every episode, a goal is selected from the three, and remains constant for the duration of the episode. Goal positions act as task indices here. For a visualisation of the environment layout, see Figure 10

Our value function network is given by the goal-dependent parameterisation: $V(s, g) = f_a(\psi(s) \odot w(g))$ ψ is a MLP with two hidden layers of size 256, and an output size of 64. This relatively small output size was chosen in order to encourage compression in the learned feature space. w is a MLP with a single hidden layer of size 128, and output of 64 to perform elementwise multiplication with the output of ψ. f_a is then a final MLP with a single hidden layer of size 64, and is trained to output the value of the state for the current task.

During this phase, we train for 20 million time steps, though this much data is not necessary. The value loss remains essentially constant after 500 000 steps. We note that it is not necessary to solve the task, or even learn the optimal value function in this phase, we only need to learn the features that correspond well with the values.

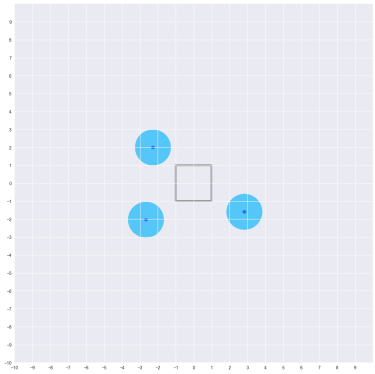


Figure 10: Goal positions during pretraining of Ant. Dark blue dots represent goal positions, light blue circles represent the radius at which the ant receives sparse reward and the episode terminates, if the corresponding goal is active. The grey box is for scale comparison with the trace plots in the paper and here.

We ran this phase using five random seeds and chose the embedding to distil randomly among them. There was no observable differences between random seeds, they learned consistently, and the saliency maps of the value function were similar for all seeds.

F.2 Distillation

The distillation phase consists of training a smaller network with a tight bottleneck to predict the output of the learned embedding ψ from the state. This is done to ensure that the absolute minimal amount of information from the state is encoded in our final learned embedding, $\bar{\psi}$. The distillation was trained on states from the same distribution as the original policy was trained on. We employed an experience replay buffer of size 1 million, in order to achieve greater data efficiency during this phase.

Our distilled network was a two hidden layer MDP with 256 units in each layer, and a bottleneck size of 8. Smaller bottleneck sizes were tried, though these led to poor saliencies after learning. The output of this bottleneck was then passed through a final linear layer, giving a 64 dimensional output. The distilled network was then trained by regression of this output against the output of ϕ, using mean-squared error loss. L1 regularisation was applied to the embedding output, and weight decay regularisation was applied to the network weights, in order to ensure the most terse and well-generalising model possible. Training was performed for 5 million time steps. Four random seeds were used, and again no significant differences were found across random seeds in terms of MSE or saliencies of learned feature maps.

F.3 Skill Learning

In this phase we learn skills that are diverse in either our learned embedding from the previous phase (for DIVRS), from the original state space (DIAYN), or a hand constructed x-y space (DIAYN + XY).

Skills were trained using SAC (Haarnoja et al., 2018) with the "double Q trick" for 3 million time steps. When learning policies and values for skills on replay buffer data, the reward function was taken from the current discriminator parameters, rather than those from when the data was collected.

For policy and value networks that condition on the active skill, a one-hot representation of the active skill is passed to a small MLP with a single hidden layer of size 64 and output dimension of 32. This embedding is concatenated with the environment observation, and passed through a three hidden-layer MLP, with the first two hidden layers of size 300, and the last of size 64, before mapping to the action space of dimension 16 for the policy network, or to a Q-value, in the case of the value network.

For reverse MI experiments, the discriminator is a two hidden layer MLP with 300 units per hidden layer. It takes in the state or embedding of state, and maps to logits for the probability that each skill is active in that state. Forward MI experiments used a lookup table corresponding to a state in the relevant space for each skill.

Five seeds were used for each skill learning procedure. Hyperparameters were selected to optimize pseudo-reward at the 3 million timestep mark. Skills were fairly consistent across seeds, though they differed somewhat in position and orientation of particular skills.

F.4 Quantitative skill Evaluation

For quantitative evaluation of skills, we employed a sparse sequential HRL learning task, in which skills had to be composed to achieve rewards. In order to keep the state Markov, we include a one hot encoding of the number of goals reached, concatenated with the agent observation space as input to the SMDP-level policies, as well as to the flat baseline agents. Training was done for 5 million time steps. HRL agents were learned using a variant of SAC designed for discrete action spaces. This version allows for explicit computation of entropy terms, rather than the sample-based version used in continuous control.

Due to the combinatorial nature of random seeding across the several phases, for these experiments, we did not select for skills across the random seeds in the skill learning phase directly. We trained across two random seeds for 12 different hyperparameter and skillset combinations. We found that the hyperparameter ranges that we employed generally did not change learning significantly. For DIVRS, we report the average performance across hyperparameters and seeds and for baselines report the (more favorable) average of the top three most successful hyperparameter configurations, which themselves are determined by the average performance across the two seeds.

The networks used for the SMDP level policy consisted of two hidden layers of size 128, followed by a hidden layer of size 64, before outputting logits for the skill selection distribution, or for the Q values for each skill.

G Pixel Reacher Experiment Details

Here we provide implementation details of the experiments performed in the Cooking Reacher environment. As was done in the Ant environment, the gear ratio of the simulated motors was lowered, this time to 80. ReLU activation functions were used for all networks, and Adam was the optimiser.

G.1 Pretraining

The pretraining procedure here is similar to the one employed in the Ant environment. First our agent is trained to reach three fixed goals in flavour space. Goal information is passed to the agent as a separate feature from the pixel state, so that our learned embedding space is not goal-dependent. Pixel information is processed via a two layer convolutional network, with a fixed kernel size of $(3, 3)$, no padding, and a stride of 2. The first layer has 64 channels, while the second has 32. From here, there is a fully connected layer of size 64. This network all together forms ϕ. The output this is elementwise multiplied by the goal embedding, which is identical to w in the Ant experiments, though now the input is in $[0, 1]^2$ to correspond with the flavour space. This phase lasts 2m frames.

G.2 Distillation

Distillation follows identically to the process described in the Ant section, but instead of an MLP, the convolutional network described in the previous section is employed. Again, a bottleneck of size 8 was used. Distillation was done for 3m frames.

G.3 Skill Learning

As in the Ant environment, the one-hot skill index is passed through a small MLP. The resulting embedding is concatenated with the output of our pixel observation CNN, which has the same architecture as the ones used in previous phases, though here the hidden layer is of size 300. In the case of the learned Q function, the action is concatenated here as well. From here, both the critic and policy networks have a single fully connected layer of size 300, which is then mapped to either a predicted value, or a distribution over actions. For the DIAYN baseline, the discriminator is identical to the network described here, though the skill id is not concatenated.

H MuJoCo Hyperparameters

Pretraining	Value
Environment Steps per Iteration	40
Environment Threads	64
GAE λ	0.99
Learning Rate	$8.96e-5$
Batches per Iteration	4
Gradient Clipping Norm	0.5
PPO Clipping Epsilon	0.2
Batch Size	1280
Learning Skills	**Value**
Learning Rate	$3e-4$
Environment Steps per Iteration	1000

I Related Work in Skill Discovery

Hierarchical Reinforcement Learning Skills are often learned as part of a hierarchical policy. To learn a useful and diverse set of skills, one can rely on specified subgoals for each skill (Sutton et al., 1999; Kulkarni et al., 2016), which can be found analyzing the structure of the MDP (McGovern & Barto, 2001; Şimşek & Barto, 2009), previous policies (Goel & Huber, 2003; Tessler et al., 2017) or predictability (Harutyunyan et al., 2019). Predictability, in contrast to discriminability, is the notion that skills should *terminate* in a predictable state, instead of *behave* in a way that is different from other skills. Subgoals can also be generated by a higher-level policy, either set in (Nachum et al., 2018a) or in a latent space with learned semantics (Vezhnevets et al., 2017; Nachum et al., 2018b).

Representation Learning Our work is also related to representation learning, in particular the learning of state-abstractions (Abel et al., 2020; Li et al., 2006; Abel et al., 2016; Jong & Stone, 2005; Igl et al., 2019a). However, unlike this prior work, our state-abstraction is only value-relevant, not action-relevant. In other words, our compressed state-representation should contain information with which progress on tasks can be measured, but *doesn't* necessarily contain information which allows for progress on tasks. For example, while in the ant-task, the x-y location is sufficient to approximately determine the value-function, an optimal policy would also require the location and orientation of various joints - exactly the type of information we aim to remove from our representation. Similarly, while (Abel et al., 2019) also utilizes an information theoretic trade-off between compression and quality of the representation, their performance evaluation captures the quality of the policy, while we only capture the quality of value predictions. Silver et al. (2017) and Oh et al. (2017) also learn a state-abstraction, but aim to capture even more information in the latent space, by also predicting temporal dynamics. On the other hand, Liu et al. (2020) and Goyal et al. (2019) capture only the task-goal in the latent space, not information relevant to measuring progress w.r.t this goal. Similar to our work, Sermanet et al. (2018) use a learned representation to generated policy rewards. However, unlike our work, they utilize a metric-learning loss to learn the representation and apply their method towards viewpoint independence, not skill discovery.