Unsupervised Skill Discovery for Learning Shared Structures across Changing Environments

Sang-Hyun Lee [1] **Seung-Woo Seo** [1]

Abstract

Learning shared structures across changing environments enables an agent to efficiently retain obtained knowledge and transfer it between environments. A skill is a promising concept to represent shared structures. Several recent works proposed unsupervised skill discovery algorithms that can discover useful skills without a reward function. However, they focused on discovering skills in stationary environments or assumed that a skill being trained is fixed within an episode, which is insufficient to learn and represent shared structures. In this paper, we introduce a new unsupervised skill discovery algorithm that discovers a set of skills that can represent shared structures across changing environments. Our algorithm trains incremental skills and encourages a new skill to expand state coverage obtained with compositions of previously learned skills. We also introduce a skill evaluation process to prevent our skills from containing redundant skills, a common issue in previous work. Our experimental results show that our algorithm acquires skills that represent shared structures across changing maze navigation and locomotion environments. Furthermore, we demonstrate that our skills are more useful than baselines on downstream tasks.

1. Introduction

Most real-world tasks require an agent to handle continuously changing environments. While humans can handle these environments by leveraging previously obtained knowledge to quickly adapt to new environments, conventional reinforcement learning (RL) agents must learn from scratch whenever an environment is changed, requiring far more data than humans. The most straightforward way to retain knowledge obtained from previous environments is to save a dataset or an independent model for each environment. However, these approaches are inappropriate to transfer knowledge between environments. A promising alternative approach to transfer knowledge is to learn and leverage shared structures across environments (Thrun & O'Sullivan, 1996; Griffiths et al., 2019). The idea of leveraging shared structures to transfer knowledge is inspired by the observation that humans address a complex task by decomposing it into simpler sub-tasks and then combining their solutions (Khetarpal et al., 2022).

A skill, also known as a temporally extended action, is an efficient concept to represent shared structures between environments (Thrun & Schwartz, 1994; Tessler et al., 2017; Schaul et al., 2018). Several recent works proposed unsupervised skill discovery algorithms that discover useful skills without a reward function (Eysenbach et al., 2018; Sharma et al., 2019; Campos et al., 2020; Liu & Abbeel, 2021b;a; Shafiullah & Pinto, 2022). They demonstrated that their skills represent consistent and distinct behaviors and can be reused to accelerate learning on downstream tasks. However, their works focused on discovering skills in stationary environments or kept a skill being trained within an episode, which is insufficient to learn or represent shared structures.

In this paper, we propose a new unsupervised skill discovery algorithm that discovers skills that can represent shared structures across changing environments. We represent shared structures as compositions of skills and hypothesize that in order to obtain such skills, we must learn the skills that retain previously obtained knowledge and that their compositions maximize state coverage in changing environments. To implement this idea, our algorithm learns incremental skills and encourages a new skill to represent distinct and consistent behaviors that expand the state coverage constructed with the compositions of previously learned skills. Figure 1 shows the overview of our algorithm.

We also introduce a skill evaluation process to prevent our skills from containing redundant skills, which is a common degenerate case in previous work. Our evaluation process decides whether to retain a new skill based on similarities between previously acquired skills and how much the new

[1]Department of Electrical and Computer Engineering, Seoul National University, Seoul, South Korea. Correspondence to: Seung-Woo Seo <sseo@snu.ac.kr>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

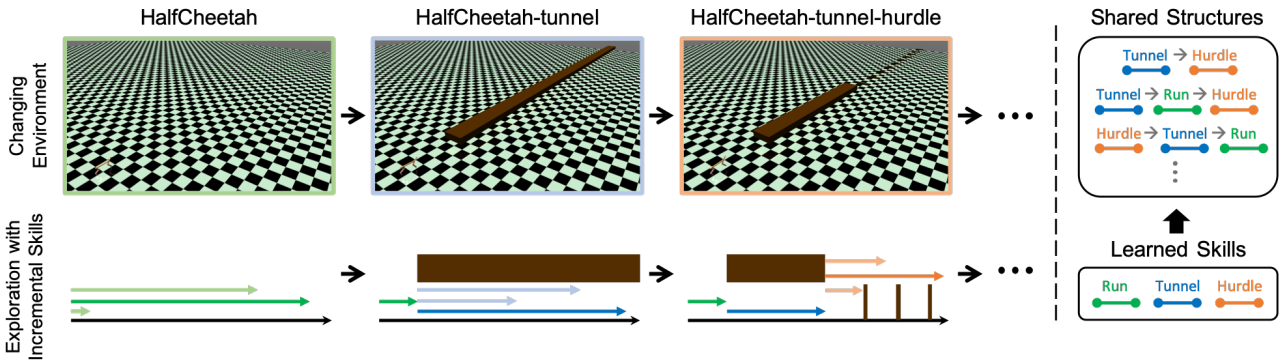

Figure 1. Overview of our unsupervised skill discovery algorithm. We design our algorithm to discover skills that can represent shared structures across changing environments. Our algorithm trains skills in an incremental fashion and encourages a new skill to expand the state coverage constructed with compositions of previously learned skills. The arrows denote learned skills and their colors denote the environments in which the skills are discovered. The darker-colored arrows denote the skills that maximize state coverage.

skill contributes to expanding state coverage. The similarities between skills are estimated by leveraging Successor Features (SFs) that encode the behaviors induced by skills as multidimensional variables (Barreto et al., 2017). In changing environments where data streams are commonly much larger than what an agent can retain (Schaul et al., 2018), our evaluation process can improve the scalability of unsupervised skill discovery algorithms.

The main contributions of our work are three-fold: 1) we introduce a novel unsupervised skill discovery algorithm that learns a set of skills that can represent shared structures across changing environments, 2) we present a skill evaluation process to ensure that recursive extensions of a skill set improve its usefulness with respect to shared structure representation, and 3) we design new changing environments and evaluate our algorithm against baselines including state-of-the-art unsupervised skill discovery algorithms. The experimental results demonstrate that our skills can represent the shared structures across changing maze navigation and locomotion environments, retaining previously obtained knowledge. We also observe that our skills are useful to accelerate learning on downstream tasks.

2. Related Work

Continual RL (CRL) is a paradigm where an agent continually learns a sequence of tasks while leveraging previously acquired knowledge. CRL algorithms can be broadly divided into three categories: explicit knowledge retention, learning to learn, and leveraging shared structure (Khetarpal et al., 2022). Explicit knowledge retention algorithms save an independent model or dataset obtained for each task (Rusu et al., 2016; Isele & Cosgun, 2018; Rolnick et al., 2019). It is the most straightforward way to prevent catastrophic forgetting in continual settings. However, it causes inefficient storage utilization and hinders an agent from utilizing knowledge obtained from previous tasks. Learning to learn algorithms, also called meta-learning algorithms, seek to improve an agent's own learning process. These algorithms generally train a neural network to represent the learning process itself or optimize the initialization parameters for fast fine-tuning (Duan et al., 2016; Finn et al., 2017; Nagabandi et al., 2018). Unlike the above two categories of CRL algorithms, leveraging shared structure algorithms attempt to learn shared structures across tasks and reuse them to adapt to new downstream tasks (Devin et al., 2017; Frans et al., 2017; Tessler et al., 2017). Our work falls into this category and seeks to represent shared structures across changing environments with a set of skills learned without a reward function.

Discovering skills without a reward function has been an active research area in the context of RL. VIC (Gregor et al., 2016) discovered skills that allow an agent to have the most control over an environment by maximizing the mutual information between final states and skills. Similarly, DIAYN (Eysenbach et al., 2018) and DADS (Sharma et al., 2019) maximized the mutual information between individual states and skills to learn skills that represent diverse and consistent behaviors. EDL (Campos et al., 2020) presented theoretical and empirical evidence that skills learned with the above algorithms offer poor state space coverage due to an insufficient exploration issue. APT (Liu & Abbeel, 2021b) and APS (Liu & Abbeel, 2021a) demonstrated that particle-based entropy maximization can be a solution to handle the limitation. Whereas these previous works focused on discovering skills in stationary environments, our work seeks to discover skills in non-stationary environments in which environment dynamics change over time. Most closely re-

lated to our work is DISk (Shafiullah & Pinto, 2022). DISk learned diverse and distinct incremental skills in changing environments without forgetting previously learned skills. However, unlike this work, our algorithm discovers skills that can represent shared structures across changing environments. Furthermore, while DISk retains all skills learned in past environments, we introduce the skill evaluation process to prevent saving redundant skills.

Barreto et al. (2017) introduced the concept of SFs and demonstrated that the SFs can be a major breakthrough for transfer in RL. The Option Keyboard (Barreto et al., 2019) used SFs to combine known options to create new options without additional learning. Barreto et al. (2018) also showed that SFs provide an efficient way to carry out Generalized Policy Improvement (GPI) and Generalized Policy Evaluation (GPE). These generalized operators allow an agent to decompose complex problems into simpler multiple subtasks. In addition to transfer in RL, SFs have been actively studied in a variety of ways. Ramesh et al. (2019) interpreted the cluster centers as landmark states or subgoals by clustering the SFs of rollout states. Machado et al. (2020) proposed the count-based exploration algorithm that uses the norm of learned SFs as an exploration bonus. Unlike these works, our method utilizes SFs to estimate similarities between learned skills.

3. Preliminaries

3.1. Markov Decision Process (MDP)

The MDP is a framework for sequential decision-making problems, which can be represented as the tuple $(S, A, P, R, \rho_0, \gamma, T)$. S and A are the set of states s and actions a, respectively. $P : S \times A \times S \to \mathbb{R}^+$ represents the state transition model. $R : S \times A \to \mathbb{R}$ is the reward function, $\rho_0 : S \to \mathbb{R}^+$ is the initial state distribution, γ is the discount factor, and T denotes the horizon. An agent takes an action sampled from a policy $\pi : S \to P(A)$, which maps states to a probability distribution over actions. The agent's goal is to find the optimal policy π^*that maximizes the expected cumulative rewards.

RL is an approach to achieving this goal when the model of environments is not known. One of the key elements in RL is the state-action value function $Q^\pi(s, a)$. The state-action value function is the expected return value obtained when the agent takes the action a in the state s and follows the policy π. This can be written as follows:

$$Q^\pi(s,a) = \mathbb{E}_\pi\Big[\textstyle\sum_{i=t}^{\infty}\gamma^{i-t} r_{i+1} | S_t = s, A_t = a\Big].$$

The state-action value function is utilized in most RL to find the optimal policy by deriving $\mathrm{argmax}_{a_t \in A} Q(s_t, a_t)$, which selects the action a_t that maximizes the expected return from state s_t.

3.2. Successor Features (SFs)

Suppose that the reward function is a linear combination of features $\phi(s, a, s') : S \times A \times S \to \mathbb{R}^d$ and weight $\mathbf{w} \in \mathbb{R}^d$, which can be written as $r(s, a, s') = \phi(s, a, s')^\top \mathbf{w}$. The weight $\mathbf{w}$ is also called a task vector, as it reflects preferences for each feature component. Now we can rewrite the state-action value function as follows:

$$\begin{aligned} Q^\pi(s,a) &= \mathbb{E}_\pi\Big[\textstyle\sum_{i=t}^{\infty}\gamma^{i-t}\phi_{i+1}^\top \mathbf{w} | S_t = s, A_t = a\Big] \\ &= \mathbb{E}_\pi\Big[\textstyle\sum_{i=t}^{\infty}\gamma^{i-t}\phi_{i+1} | S_t = s, A_t = a\Big]^\top \mathbf{w} \\ &\equiv \psi^\pi(s,a)^\top \mathbf{w}. \end{aligned}$$

Barreto et al. (2017) called $\psi^\pi(s, a)$ the Successor Features (SFs) of (s, a) under policy π, as they interpreted the SFs as a generalization of the Successor Representation (SR). The SFs $\psi^\pi(s, a)$ represent the expected discounted sum of features $\phi(s, a, s')$ when following policy π in a given environment. The SFs can then be regarded as a multidimensional value function with rewards $\phi(s, a, s')$, which means that they can be trained using standard RL algorithms. In this work, we use SFs to estimate the similarities between the retained skills and the new skill.

4. Proposed Method

We consider an unsupervised RL problem in changing environments. This problem consists of two phases. In the first phase, an agent seeks to obtain useful knowledge from changing environments with reward-free interactions. These reward-free interactions are assumed to be inexpensive so that the agent can freely interact with the environments. During the second phase, knowledge obtained from the previous phase is evaluated based on how much it helps the agent adapt to downstream tasks. As we discussed in Section 1, shared structures across changing environments are an efficient form to retain and transfer knowledge.

Similar to previous skill discovery algorithms, we encode skills as latent variables $z \in Z$ and design independent skill policies $\pi_{z_i}(a|s)$ for each skill. The skill policies take the state as input and output primitive action at every step. For brevity of notation, we denote the skill policies as $\pi_i(a|s)$.

4.1. Discovering Skills for Learning Shared Structures across Changing Environments

The shared structures between changing environments can be represented in various forms. Our goal is to discover skills that can represent the shared structures in their compositions. We hypothesize that to discover such skills, we should learn skills that retain previously obtained knowledge and maximize state coverage with their compositions. Building on this idea, our algorithm learns skills incremen-

tally and defines a master policy $\pi(z|s)$ that encourages a new skill to represent behaviors that can expand the state coverage constructed with previously learned skills. The master policy selects a skill and the skill is kept for a fixed number of steps. Unlike the skill policies that should represent distinct and consistent behaviors to be reusable across environments, the master policy should be able to adaptively combine skills to explore the environment encountered. Note that the previously learned skill policies are fixed when we learn a new skill policy.

Here we describe our objective. In our objective, both the new skill policy and the master policy maximize the entropy of states $\mathcal{H}(S)$. This term encourages the new skill policy to represent distinct behaviors from previously learned skills. At the same time, this term encourages the master policy to compose skills to maximize state coverage. Since previously learned skills had been trained to represent behaviors maximizing coverage of the past environments, the master policy will choose these skills until new behaviors are required to expand coverage in the environment encountered.

The master policy is also trained to maximize the entropy of skills conditioned on states $\mathcal{H}(Z|S)$, which allows an agent to select diverse skills including the new skill when the agent has no obvious skills to efficiently expand state coverage. This helps the new skill policy represent behaviors expanding state coverage as it is trained to represent distinct behaviors from previously acquired skills. In addition, the skill policy minimizes the entropy of the next states conditioned on states and the current skill $\mathcal{H}(S'|S,Z)$ because it should represent consistent behaviors to be reusable. As a result, given the previously learned skill policies $\pi_{1:m-1}$, our objective can then be written as follows:

$$\mathcal{F}(\theta) \triangleq \mathcal{H}(S) + \mathcal{H}(Z|S) - \mathcal{H}(S'|S, Z = z_m) \quad (1)$$

where z_m is a new skill being trained. Appendix A describes the distinguishing features of our objective.

While the second term in our objective can be optimized with MaxEnt RL algorithms, the other two terms are intractable to compute, as we have no access to the true distribution of states and the dynamics. We address this issue by utilizing practical techniques to estimate both terms. First, we maximize the third term by deriving its variational lower bound as follows:

$$\begin{aligned}
&-\mathcal{H}(S'|S, Z = z_m) \\
&\quad = \textstyle\sum_{s,s'} p(s', s|z_m) \log p(s'|s, z_m) \\
&\quad = \mathbb{E}_{s,s'\sim\pi_m}[\log q(s'|s, z_m)] + \mathbb{E}_{s\sim\pi_m}[D_{KL}(p\|q)] \\
&\quad \geq \mathbb{E}_{s,s'\sim\pi_m}[\log q(s'|s, z_m)], \qquad (2)
\end{aligned}$$

where we use the non-negativity of KL divergence and introduce $q(s'|s,z)$ as a variational approximation of the true transition function $p(s'|s,z)$.

Next, we approximate the first term $\mathcal{H}(S)$ with the nonparametric particle-based entropy estimator (Singh et al., 2003), similar to (Liu & Abbeel, 2021b; Shafiullah & Pinto, 2022). The key concept behind this estimator is to measure the sparsity of the distribution based on the distance between each particle and its k nearest neighbor. To be specific, given N samples $\{x_i\}_{i=1}^N \sim p(X)$ defined on a q-dimensional space $X \in \mathbb{R}^q$, the particle-based approximation for a distribution $p(X)$ can then be written as follows:

$$\hat{\mathcal{H}}_{k,\mathbf{X}}(p) = -\frac{1}{N}\sum_{i=1}^{N} \ln \frac{k\Gamma(q/2+1)}{N\pi^{q/2}R_{i,k,\mathbf{X}}^q} + b(k),$$

where Γ is the gamma function, $b(k)$ is a bias correction term, and $R_{i,k,\mathbf{X}} = \|x_i - x_i^{(k)}\|$ is the Euclidean distance between particle x_i and its k^{th} nearest neighbor $x_i^{(k)}$. We can simplify this approximation by ignoring the terms independent of x_i as follows:

$$\hat{\mathcal{H}}_{k,\mathbf{X}}(p) \propto \sum_{i=1}^{N} \ln \|x_i - x_i^{(k)}\|. \quad (3)$$

To make the distance between particles meaningful for representing shared structures, we incorporate our inductive bias that the skill policies should be shared across environments, and the master policy should be able to combine skills differently depending on environments. To encode this bias, we define two independent mapping functions $\sigma_S(s)$ and $\sigma_M(s)$ for the skill policies and the master policy, respectively. The former maps states to agent-specific representations such as the agent's velocity and the latter maps states to environment-specific representations such as the positions of obstacles.

Based on the above approximations, we can train both the master policy and the new skill policy in a reinforcement-learning style. The master policy can be trained to maximize the first and the second terms in our objective with the intrinsic rewards $r_M(s,a,s')$ defined as follows:

$$r_M(s,a,s') = \|\sigma_M(s') - \sigma_M(s')^{(k)}\| - \log \pi(z|s). \quad (4)$$

Similarly, the new skill policy can be trained to maximize the first and third terms in our objective with the intrinsic rewards $r_S(s,a,s')$ defined as follows:

$$r_S(s,a,s') = \|\sigma_S(s') - \sigma_S(s')^{(k)}\| + \log q(s'|s, z_m). \quad (5)$$

4.2. Saving Discovered Skills with Evaluation Process

A common degenerate case in unsupervised skill discovery algorithms is that skills represent static behaviors or are distinguished by small state differences. This issue is exacerbated when the dynamics of environments is complicated.

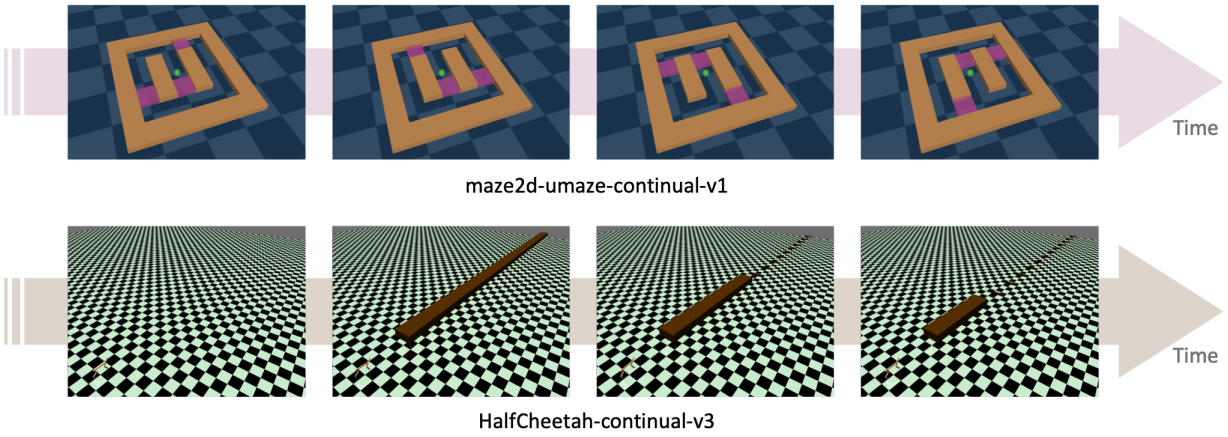

Figure 2. Changing environments introduced in our work. The shared structures across both changing environments can be represented as compositions of skills. **Top:** maze2d-umaze-continual-v1. The positions of the three purple boxes change over time. **Bottom:** HalfCheetah-continual-v3. The obstacles added to the environment change over time.

In our settings, such degenerate skills may retain redundant knowledge to represent shared structures, and we need to prevent these degenerate skills from being included in our skills. To this end, we introduce a skill evaluation process that determines whether to retain a new skill. Our evaluation process is designed to ensure that recursive expansions of a skill set improve its usefulness in terms of representing shared structures between changing environments.

We evaluate the usefulness of a skill on the following two criteria. First, the set of skills including the new skill should provide wider state coverage than that without the new skill. This can be formulated as whether the state entropy induced by the compositions of skills increases as the new skill is added. Second, a new skill should not be similar to the skills already stored. For this, we use SFs to encode the dynamics induced by each skill as a multidimensional variable. The SFs of (s, a) under a skill policy π_i are defined as follows:

$$\psi^{\pi_i}(\bar{s}, a) = \mathbb{E}_{\pi_i}[\sum_{k=t}^{K} \gamma^{k-t} \phi_{k+1} | S_t = \bar{s}, A_t = a],$$

where K is the number of steps in which a skill is kept and the input state $\bar{s}$ of SFs is the state s concatenated with the ratio between t and K as a skill can be initiated at any state. This allows us to estimate the similarity between two skills as the cosine similarity S_C between their SFs as follows:

$$S_C(\psi^{\pi_i}(\bar{s}, a), \psi^{\pi_j}(\bar{s}, a)) = \frac{\psi^{\pi_i}(\bar{s}, a)^\top \psi^{\pi_j}(\bar{s}, a)}{\|\psi^{\pi_i}(\bar{s}, a)\| \|\psi^{\pi_j}(\bar{s}, a)\|}. \quad (6)$$

Now, we can formalize the second criterion as whether the maximum similarity between the new skill and previously learned skills is less than a certain threshold. Note that SFs of previously learned skills are retrained in the environment where the new skill is added. This is because behaviors induced by them can be changed due to changes in environments. SFs can be optimized with any RL algorithm as we discussed in Section 3. To sum up, given the previously acquired skills $z_{1:m-1}$, the criteria to save a new skill z_m can be written as follows:

$$\textbf{Criterion 1.} \quad \hat{\mathcal{H}}_{1:m-1}(S) \leq \hat{\mathcal{H}}_{1:m}(S) \quad (7)$$

$$\textbf{Criterion 2.} \quad \max_{1 \leq i \leq m-1} \mathbb{E}_{\pi_{1:m}}[S_C(\psi^{\pi_m}, \psi^{\pi_i})] \leq \eta \quad (8)$$

where $\hat{\mathcal{H}}_{1:m}(S)$ is the estimated entropy of states obtained with $\pi(z|s)$ and $\pi_{1:m}(a|s)$, and η is the similarity threshold. Our evaluation process is repeated whenever a new skill is added. Once a new skill is saved, it is fixed and then used to evaluate subsequent skills. Appendix B describes our algorithm's overall training procedure in further detail.

5. Experiments

Our experiments aim to answer the following questions: (1) Can our algorithm discover a set of skills that represent shared structures across changing environments? (2) Can our skill evaluation process prevent degenerate skills from being included in a skill set? (3) Can our skills accelerate the learning of downstream tasks? To answer these questions, we introduce new changing maze navigation and locomotion environments called maze2d-umaze-continual-v3 and HalfCheetah-continual-v3. As described in the top row of Figure 2, the maze2d-umaze-continual-v1 is an extension

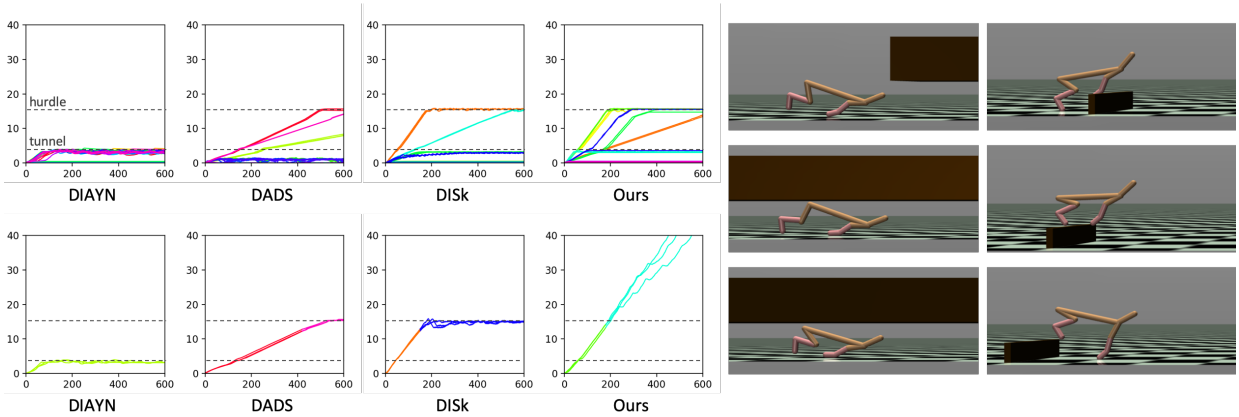

Figure 3. Skills learned for HalfCheetah-continual-v3. The environment that the skills are visualized has a tunnel section from 4.0 and a hurdle section from 16.0, denoted as dotted lines in the plots. **Left:** Trajectories of learned skills. The top row shows the X positions of the agent over time for each skill, and the bottom row shows the X positions of the agent over time for skills composed to maximize state coverage as in our algorithm. In contrast to baselines, our algorithm learned skills that can handle hurdles. **Right:** Visualization of our skills for a tunnel and hurdles. These skills are depicted by green and cyan lines in the rightmost plot of the bottom row.

of maze2d-umaze-v1 from D4RL (Fu et al., 2020). There are three boxes and their positions change over time. We learn three skills for each configuration of the boxes. An agent is spawned at the center of the maze and the area the agent can explore depends on the positions of the boxes. The shared structures across the maze environment are connections between passages leading up, down, left, and right, which cannot be represented with a single skill.

As described in the bottom row of Figure 2, HalfCheetah-continual-v3 is a variant of HalfCheetah-v3 provided by OpenAI Gym (Brockman et al., 2016). We add a tunnel and hurdles as obstacles to the environment, and the added obstacles change over time. We discover three skills for each configuration of the added obstacles. The behaviors required to pass through the two sections are entirely opposite to each other. The agent must jump to cross the hurdles, while it must crawl with its head down to get through the tunnel. This prevents the agent from passing through both sections with a single skill. The shared structures across the changing environment are connections between run, hurdle, and tunnel sections. Appendix C provides further details on our environments and experimental setup.

We would like to emphasize that to represent the shared structures of changing environments with skills, we must obtain skills that allow an agent to explore every component of the shared structures. This poses a significant challenge to existing unsupervised skill discovery algorithms that suffer from catastrophic forgetting or assume a skill is fixed within an episode in the training phase. Note that in the figure of this section, we encode the order in which skills are saved in the rainbow color map ranging from red to purple.

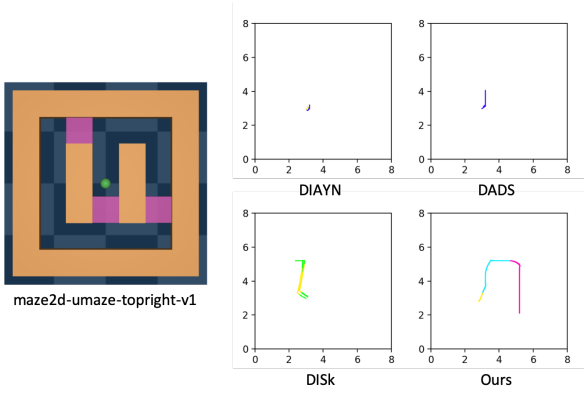

Figure 4. **Left:** maze2d-umaze-topright-v1. Skills are visualized in this environment that an agent encountered in the past. **Right:** Trajectories generated by compositions of skills learned for maze2d-umaze-continual-v1. Skills learned with baselines are combined to maximize state coverage as in our method. The composition of our skills represents the structure of this maze.

5.1. Can Our Skills Represent Shared Structures across Changing Environments?

Here we demonstrate that our algorithm can learn skills that represent shared structures across changing environments without a reward function. We evaluate our algorithm against state-of-the-art unsupervised skill discovery algorithms DIAYN, DADS, and DISk in HalfCheetah-continual-v3 and maze2d-umaze-continual-v1. To be a fair comparison, all baselines were implemented with independent skill networks as in our algorithm. A Comparison with DIAYN and DADS shows how important it is to retain knowledge ob-

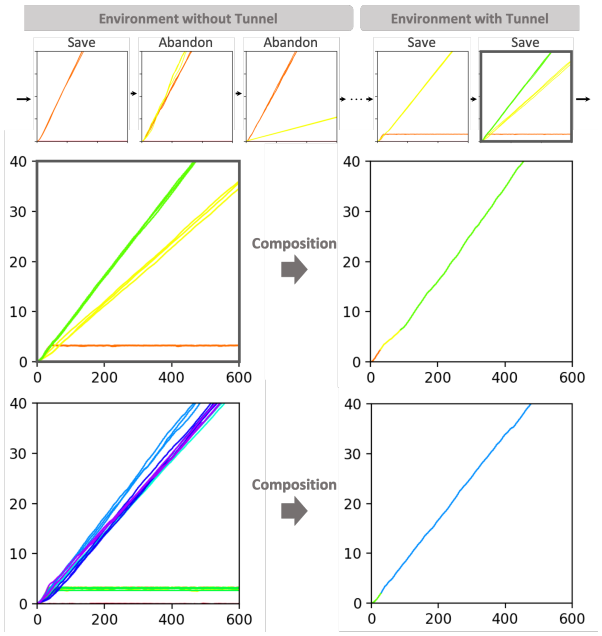

Figure 5. Analysis of our skill evaluation process in HalfCheetah-continual-v3. The plots show the X position of the agent over time. **Top:** Skills learned with our evaluation process. These skills do not include redundant skills and their composition achieves state coverage comparable to that of all skills. **Bottom:** Skills learned without our evaluation process.

tained from previous environments for representing shared structures, while a comparison with DISk shows how important it is to learn a new skill that expands the state coverage constructed with compositions of previously learned skills. Please refer to Section 2 for further details about baselines.

We visualize the trajectories of the skills learned for HalfCheetah-continual-v3 in Figure 3. The environment in which the skills are visualized is the past environment including a tunnel section from 4.0 to 12.0 and a hurdle section from 16.0. The top row in Figure 3 shows each learned skill's trajectories of an agent within an episode. Since a skill cannot represent behaviors that can handle both a tunnel and hurdles, we observe that all the skills represent behaviors that cannot even go through a tunnel or can go through a tunnel but not jump over hurdles. The bottom row of Figure 3 shows the trajectories generated by composing skills as in our algorithm. While the composition of the skills learned with baselines still cannot generate behaviors that can explore the hurdle section, the composition of our skills generates flexible behaviors that can handle both the tunnel and the hurdle sections. These results demonstrate that discovering skills that can expand state coverage is important to represent shared structures with learned skills. We visualize our skills that can handle a tunnel or hurdles on the right side of Figure 3.

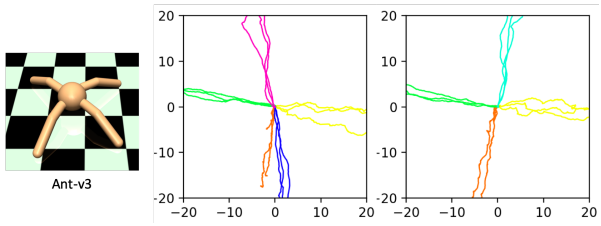

Figure 6. Analysis of our skill evaluation process in Ant-v3. **Left:** Skills learned without our evaluation process. They consist of all skills colored from red to purple, including redundant skills that represent static or similar behaviors. **Right:** Skills learned with our evaluation process. Although these skills consist of fewer skills colored from red to cyan, they represent diverse behaviors comparable to those represented by all the skills.

Figure 4 shows the trajectories of learned skills on maze2d-umaze-continual-v3. The trajectories are generated by composing skills to maximize state coverage in the environment described on the left. An agent had already encountered this environment for the second time. DIAYN and DADS fail to learn skills that represent the structure of this environment. It's because all of their skills are re-trained in the last maze environment, described in the rightmost column in Figure 2. This causes their skills to forget previously obtained knowledge. We empirically observe that their skills are mostly directed downwards and only a few move upwards at very low speeds as one box is right above the agent's starting position in the last environment. In contrast, our algorithm successfully represents the structure of this maze without forgetting previously learned skills. DISk fails to represent the structure with skills even though it can retain previously obtained skills. These results indicate that learning the skills allowing an agent to expand state coverage is important to obtain skills that can represent shared structures.

5.2. Can Our Skill Evaluation Process Prevent Our Skills from Containing Degenerate Skills?

We demonstrate that our skill evaluation process can prevent degenerate skills from being stored without compromising the usefulness of the skill set. To clarify the impact of our skill evaluation process, we have an agent discover five skills each in the first and the second obstacle configuration of HalfCheetah-continual-v3 described in Figure 2. In these consecutive environments, a skill that overlaps with previously learned skills or moves at a lower speed than them is regarded as a degenerate skill because it is not needed to expand state coverage.

Figure 5 shows how our skill evaluation process works in discovering skills. The top row depicts the skills learned with our skill evaluation process. We observe that the second and third skills discovered in the first environment are

Table 1. Average returns computed over 100 episodes on downstream tasks. These results show that our agent significantly outperforms baselines in both maze and locomotion downstream tasks. This indicates that our skills are more efficient than those learned with baselines to transfer knowledge.

METHOD	MAZE2D-MEDIUM	HALFCHEETAH-HURDLE-TUNNEL-SEQUENCE	HALFCHEETAH-TUNNEL-HURDLE-SEQUENCE
DISK	127.67 ± 60.51	88.99 ± 21.60	304.47 ± 20.44
OURS	$\mathbf{286.15 \pm 47.46}$	$\mathbf{608.69 \pm 86.85}$	$\mathbf{619.38 \pm 47.32}$
DIAYN	7.96 ± 0.79	69.03 ± 16.82	63.83 ± 5.17
DADS	22.63 ± 1.16	70.53 ± 2.04	237.90 ± 3.29

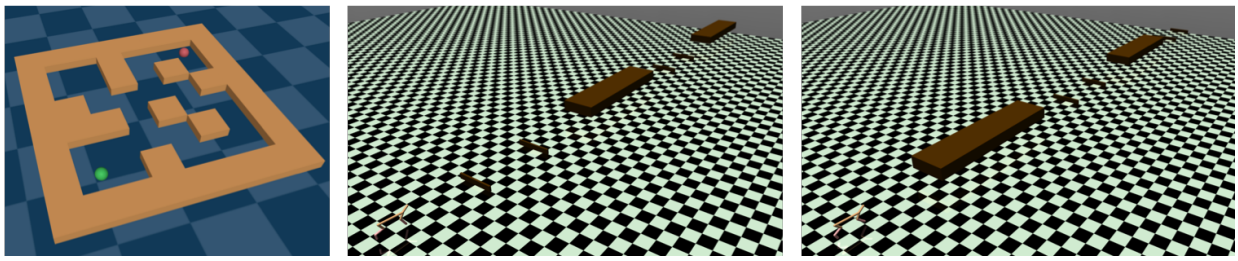

Figure 7. Downstream tasks used in our work. **Left:** maze2d-medium-v1. **Middle:** HalfCheetah-hurdle-tunnel-sequence-v3. **Right:** HalfCheetah-tunnel-hurdle-sequence-v3.

abandoned because these skills represent similar or slower behaviors than an already saved skill colored orange. The bottom row shows the skills learned without our skill evaluation process. The set of skills consists of all learned skills including redundant skills. We also notice that the state coverage obtained with the compositions of skills learned with our evaluation process is comparable to the state coverage obtained using the composition of all learned skills. These results indicate that our skill evaluation process can prevent our skills from including redundant skills without forgetting useful skills, which is important to improve the scalability of skill discovery algorithms. We provide additional analysis of our evaluation process in Appendix E.

Our skill evaluation process can also be utilized when we learn diverse and consistent skills in stationary environments. To demonstrate this, we modified our algorithm to sample skills randomly at the beginning of the episode and then discover skills with our evaluation process on Ant-v3. In this environment, an agent is spawned at the origin of an open x-y plane without obstacles. The trajectories of skills learned with and without our skill evaluation process are visualized in the left and right plots of Figure 6. We observe that the skills learned with our evaluation process represent diverse behaviors comparable to those represented by all learned skills, even though they consist of fewer skills. These results demonstrate that our evaluation process can prevent static or redundant skills from being saved when learning skills in stationary environments.

5.3. Can Our Skills Accelerate Learning on Downstream Tasks?

In this section, we show how our skills are useful on downstream tasks. Figure 7 depicts the downstream tasks used: maze2d-medium-v1, HalfCheetah-hurdle-tunnel-sequence-v3, and HalfCheetah-tunnel-hurdle-sequence-v3. The maze2d-medium-v1 provided by D4RL (Fu et al., 2020) has a more complex and larger layout than maze2d-umaze-v1 and requires an agent to reach the fixed destination. The goal of the second and third downstream tasks is to move forward as fast as possible in each environment where hurdles and tunnels alternate in a different order. An agent in these tasks can leverage skills learned in HalfCheetah-continual-v3. We use provided dense rewards to train a hierarchical agent on all downstream tasks.

Table 1 summarizes the average returns on our downstream tasks. These numerical results are computed over 100 episodes. In maze2d-medium-v1, DISk achieves relatively good performance, unlike other baselines. This implies that retaining previously learned skills is critical to transfer knowledge in changing environments. We observe that our algorithm significantly outperforms baselines on all the downstream tasks. This indicates that our skills are more efficient to retrain and transfer knowledge obtained from changing environments. Appendix F provides additional results on our downstream tasks, including learning curves and skill visualizations.

6. Conclusion

We presented a new unsupervised skill discovery algorithm that learns a set of skills that can represent shared structures across changing environments. Our algorithm learns incremental skills and encourages a new skill to expand state coverage constructed with compositions of previously learned skills. We also introduced a skill evaluation process to prevent our skill set from including degenerate skills. Our experimental results demonstrated that our skills retain knowledge from previous environments and represent shared structures across changing maze navigation and locomotion environments. We also showed that our skills allow an agent to transfer knowledge to accelerate adaptation to new downstream tasks.

There are several limitations to be addressed in future work. First, we use the predefined mapping functions to discover skills with meaningful representations, similar to previous works (Eysenbach et al., 2018; Sharma et al., 2019; Shafiullah & Pinto, 2022). While learning these functions is orthogonal and complementary to our work, we expect that combining our algorithm with state-of-the-art representation learning algorithms would be a promising research direction to alleviate this limitation (Laskin et al., 2020; Yarats et al., 2021; Laskin et al., 2022). Second, we focus on discovering skills alone without considering where to initiate or terminate them. We plan to leverage demonstrations or offline data to learn these conditions without domain knowledge (Lee & Seo, 2020; Jiang et al., 2022). Third, we assume a passive non-stationary setting where an agent cannot control the order of environments to be encountered. This setting can restrict fascinating research opportunities, such as removing previously learned skills to make the set of skills more compact. We think that discovering skills with active non-stationary settings where an agent can directly affect the non-stationary of environments can be another interesting research direction (Khetarpal et al., 2022).

Acknowledgements

This research was supported by the Challengeable Future Defense Technology Research and Development Program through the Agency For Defense Development(ADD) funded by the Defense Acquisition Program Administration(DAPA) in 2023(No.915027201), the Institute of New Media and Communications, the Institute of Engineering Research, and the Automation and Systems at Seoul National University.

References

Barreto, A., Dabney, W., Munos, R., Hunt, J. J., Schaul, T., van Hasselt, H. P., and Silver, D. Successor features for transfer in reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Barreto, A., Borsa, D., Quan, J., Schaul, T., Silver, D., Hessel, M., Mankowitz, D., Zidek, A., and Munos, R. Transfer in deep reinforcement learning using successor features and generalised policy improvement. In *International Conference on Machine Learning*, pp. 501–510. PMLR, 2018.

Barreto, A., Borsa, D., Hou, S., Comanici, G., Aygün, E., Hamel, P., Toyama, D., Mourad, S., Silver, D., Precup, D., et al. The option keyboard: Combining skills in reinforcement learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Borsa, D., Barreto, A., Quan, J., Mankowitz, D., Munos, R., Van Hasselt, H., Silver, D., and Schaul, T. Universal successor features approximators. *arXiv preprint arXiv:1812.07626*, 2018.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Campos, V., Trott, A., Xiong, C., Socher, R., Giró-i Nieto, X., and Torres, J. Explore, discover and learn: Unsupervised discovery of state-covering skills. In *International Conference on Machine Learning*, pp. 1317–1327. PMLR, 2020.

Devin, C., Gupta, A., Darrell, T., Abbeel, P., and Levine, S. Learning modular neural network policies for multi-task and multi-robot transfer. In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 2169–2176. IEEE, 2017.

Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., and Abbeel, P. Rl^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

Frans, K., Ho, J., Chen, X., Abbeel, P., and Schulman, J. Meta learning shared hierarchies. *arXiv preprint arXiv:1710.09767*, 2017.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning, 2020.

Gregor, K., Rezende, D. J., and Wierstra, D. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

Griffiths, T. L., Callaway, F., Chang, M. B., Grant, E., Krueger, P. M., and Lieder, F. Doing more with less: meta-reasoning and meta-learning in humans and machines. *Current Opinion in Behavioral Sciences*, 29: 24–30, 2019.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Isele, D. and Cosgun, A. Selective experience replay for lifelong learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Jiang, Y., Liu, E., Eysenbach, B., Kolter, J. Z., and Finn, C. Learning options via compression. *Advances in Neural Information Processing Systems*, 35:21184–21199, 2022.

Khetarpal, K., Riemer, M., Rish, I., and Precup, D. Towards continual reinforcement learning: A review and perspectives. *Journal of Artificial Intelligence Research*, 75:1401–1476, 2022.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Laskin, M., Srinivas, A., and Abbeel, P. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pp. 5639–5650. PMLR, 2020.

Laskin, M., Liu, H., Peng, X. B., Yarats, D., Rajeswaran, A., and Abbeel, P. Unsupervised reinforcement learning with contrastive intrinsic control. *Advances in Neural Information Processing Systems*, 35:34478–34491, 2022.

Lee, S.-H. and Seo, S.-W. Learning compound tasks without task-specific knowledge via imitation and self-supervised learning. In *International Conference on Machine Learning*, pp. 5747–5756. PMLR, 2020.

Liu, H. and Abbeel, P. Aps: Active pretraining with successor features. In *International Conference on Machine Learning*, pp. 6736–6747. PMLR, 2021a.

Liu, H. and Abbeel, P. Behavior from the void: Unsupervised active pre-training. *Advances in Neural Information Processing Systems*, 34:18459–18473, 2021b.

Machado, M. C., Bellemare, M. G., and Bowling, M. Count-based exploration with the successor representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 5125–5133, 2020.

Nagabandi, A., Clavera, I., Liu, S., Fearing, R. S., Abbeel, P., Levine, S., and Finn, C. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *arXiv preprint arXiv:1803.11347*, 2018.

Ramesh, R., Tomar, M., and Ravindran, B. Successor options: An option discovery framework for reinforcement learning. *arXiv preprint arXiv:1905.05731*, 2019.

Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., and Wayne, G. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Schaul, T., van Hasselt, H., Modayil, J., White, M., White, A., Bacon, P.-L., Harb, J., Mourad, S., Bellemare, M., and Precup, D. The barbados 2018 list of open issues in continual learning. *arXiv preprint arXiv:1811.07004*, 2018.

Shafiullah, N. M. and Pinto, L. One after another: Learning incremental skills for a changing world. *arXiv preprint arXiv:2203.11176*, 2022.

Sharma, A., Gu, S., Levine, S., Kumar, V., and Hausman, K. Dynamics-aware unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657*, 2019.

Singh, H., Misra, N., Hnizdo, V., Fedorowicz, A., and Demchuk, E. Nearest neighbor estimates of entropy. *American journal of mathematical and management sciences*, 23 (3-4):301–321, 2003.

Tessler, C., Givony, S., Zahavy, T., Mankowitz, D., and Mannor, S. A deep hierarchical approach to lifelong learning in minecraft. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Thrun, S. and O'Sullivan, J. Discovering structure in multiple learning tasks: The tc algorithm. In *ICML*, volume 96, pp. 489–497, 1996.

Thrun, S. and Schwartz, A. Finding structure in reinforcement learning. *Advances in neural information processing systems*, 7, 1994.

Yarats, D., Fergus, R., Lazaric, A., and Pinto, L. Reinforcement learning with prototypical representations. In *International Conference on Machine Learning*, pp. 11920–11931. PMLR, 2021.

A. Comparison to Prior Unsupervised Skill Discovery Algorithms

We would like to emphasize that our objective, $H(S) + H(Z|S) - H(S'|S, Z)$, has several distinguishing features. The key feature lies in how we utilize the first term, $H(S)$, to maximize state coverage. In contrast to previous works that maximize $H(S)$ without composing skills, we maximize it by composing skills with a high-level policy introduced in our work. Our experimental results demonstrate that this difference relieves individual skills from the burden of expanding state coverage, allowing them to focus on representing consistent and distinct behaviors. Another key feature is that while previous works typically minimize the $H(Z|S)$ term to ensure that each skill exhibits a unique behavior, we maximize this term to encourage the master policy to compose skills in diverse ways.

B. Implementation Details

B.1. Training Procedure

Algorithm 1 describes the overall procedure by which our algorithm learns a new skill policy $\pi_m(a|s)$ in an incremental fashion. Before learning the new skill policy, we fix the previously learned skills and reinitialize other models. We alternately update dynamics $q_\phi(s'|s, z)$ and both policies $\pi(z|s)$ and $\pi_m(a|s)$, which both generate learning signals for each other. While both policies can be optimized with any RL algorithm, we use Soft Actor-Critic (SAC) (Haarnoja et al., 2018) to update them in our experiments. We determine whether to retain the learned skill policy $\pi_m(a|s)$ by our skill evaluation process, described in Algorithm 2. We randomly compose skills instead of using our master policy $\pi(z|s)$ so that SFs can be trained on a variety of data. SFs are trained to minimize the temporal difference (TD) errors with SGD. Note that we train all the SFs when an environment is changed. It's because our evaluation process compares the skills based on their behaviors induced in the same environment, even if the environments in which they were trained are different. We use Adam optimizer (Kingma & Ba, 2014) to update the parameters of all models. The other hyper-parameters we used in our experiments are described in Appendix C.

Algorithm 1 Unsupervised Skill Discovery for Learning Shared Structures across Changing Environments

Given: Skill set Π
Add new skill π_m
Fix previously learned skills $\pi_{1:m-1}$
Re-initialize $\pi(z|s)$, $q(s'|s, z)$, and all $\psi^{\pi_{1:m}}(\bar{s}, a)$
while not converged **do**
  Collect N trajectories τ with $\pi(z|s)$, and $\pi_{1:m}(a|s)$
  Update $q(s'|s, z)$ using SGD to maximize Equation (2)
  Compute r_M and r_S for trajectories τ with Equation (4) and (5)
  Update $\pi(z|s)$ and $\pi_m(a|s)$ with r_M and r_S respectively using any RL algorithm
end while
Estimate state entropy $\mathcal{H}_m(S)$ for trajectories τ
$\Pi \leftarrow \Pi \cup$ SkillEvaluator$(\pi_{1:m}, \psi^{\pi_{1:m}}, \mathcal{H}_m(S))$

Algorithm 2 SkillEvaluator$(\pi_{1:m}, \psi^{\pi_{1:m}}, \mathcal{H}_m(S))$

Given: $\mathcal{H}_{m-1}(S)$
while Not converged **do**
  Collect N' trajectories τ' by composing $\pi_{1:m}(a|s)$ randomly
  Update $\psi^{\pi_{1:m}}(\bar{s}, a)$ using any RL algorithm
end while
if $H_m(S) > H_{m-1}(S)$ and $\max_i \mathbb{E}_{\pi_{1:m}}[S_C(\psi^{\pi_m}, \psi^{\pi_i})] < \eta$ **then**
  return π_m
end if
return $\{\}$

B.2. Network Architecture

Our algorithm consists of four main models: the master policy $\pi(z|s)$, the skill policies $\pi_i(a|s)$, the skill dynamics $q(s'|s,z)$, and the SFs $\psi^{\pi_i}(s,a)$. All models are represented by neural networks. The master policy and the skill policy have two hidden layers of 256 units with ReLU activations. The skill policy outputs the parameters of Gaussian distribution to handle continuous action space, and the master policy outputs the parameters of categorical distribution as we encode skills as discrete latent variables. The skill dynamics have the same network structure described in Sharma et al. (2019). Similar to both policies, SFs have two hidden layers of 256 units and each layer is followed by ReLU activations. While SFs for each skill is implemented with an independent network in our experiments, SFs of all skills also can be represented as a single shared network that takes a skill z_i as an additional input (Borsa et al., 2018).

C. Experimental Details

C.1. Environments

Our work introduces two changing environments: maze2d-umaze-continual-v1 and HalfCheetah-continual-v3. The following paragraphs provide further details about these environments.

C.1.1. MAZE2D-UMAZE-CONTINUAL-V1

We show the introduced changing environment maze2d-umaze-continual-v1 in Figure 2 of the main paper. This environment is an expansion of maze2d-umaze-v1 from D4RL (Fu et al., 2020) and consists of four sub-environments: maze2d-umaze-topleft-v1, maze2d-umaze-topright-v1, maze2d-umaze-bottomleft-v1, and maze2d-umaze-bottomright-v1. These sub-environments are constructed by changing the position of the three boxes over time. The box has a square shape measuring 0.5 on each side and restricts the agent's movement. An agent is spawned at the center of this environment [2.8, 2.8]. A state represents the position and velocity of the agent in an XY plane, and an action represents the force pushing the agent to each axis. Following previous skill discovery algorithms, we exclude the agent's position from the input of the skill policy, which makes the skill policy represent behaviors agnostic to the agent's location. The mapping function $\sigma_M(s)$ restricts the state to the agent's position, and $\sigma_S(s)$ restricts the state to the agent's velocity.

C.1.2. HALFCHEETAH-CONTINUAL-V3

Figure 2 of the main paper describes the introduced changing environment HalfCheetah-continual-v3. In this environment, we add a tunnel and hurdles as obstacles to the environment, and the configuration of added obstacles changes over time. The height and width of the hurdles are 0.3 and 0.1. We positioned the hurdles 4.0 apart. The height of the tunnel is 0.9. To ensure that an agent could not pass both sections without combining skills, we designed the agent to know the distance to the nearest forward obstacle but not its type. We empirically confirmed that none of the skills learned with baselines or our algorithm could pass both sections without being combined with other skills. We encouraged the agent to learn forward

HYPERPARAMETER	VALUE
LEARNING RATE	0.0003
DISCOUNT FACTOR	0.99
MINI-BATCH SIZE (MASTER)	128
MINI-BATCH SIZE (OTHERS)	256
ADAM β_1	0.9
ADAM β_2	0.999
TEMPERATURE	0.1
TOTAL NUMBER OF SKILLS	12
ON-POLICY SAMPLE (MAZE)	1800
ON-POLICY SAMPLE (CHEETAH)	5400
SKILL SPAN (MAZE)	10
SKILL SPAN (CHEETAH)	30
NEAREST NEIGHBOR k	3
SIMILARITY THRESHOLD η	0.85

Table 2. Hyperparameters

movements by restricting it from moving backward from its starting position. Similar to maze2d-umaze-continual-v1, the agent's position is excluded from the input of the skill policy. The mapping function $\sigma_M(s)$ maps the state to the agent's X-axis position and $\sigma_S(s)$ restricts the state to the agent's x-axis velocity.

C.2. Hyperparameters

Table 2 describes the hyperparameters used in our experiments. We used a coarse grid search to tune the hyperparameters (e.g., policy learning rate over 0.0001, 0.0003, and 0.001, mini-batch size for master policy over 32, 64, 128, and 256, and mini-batch size for skill policy over 128, 256, 512, and 1024).

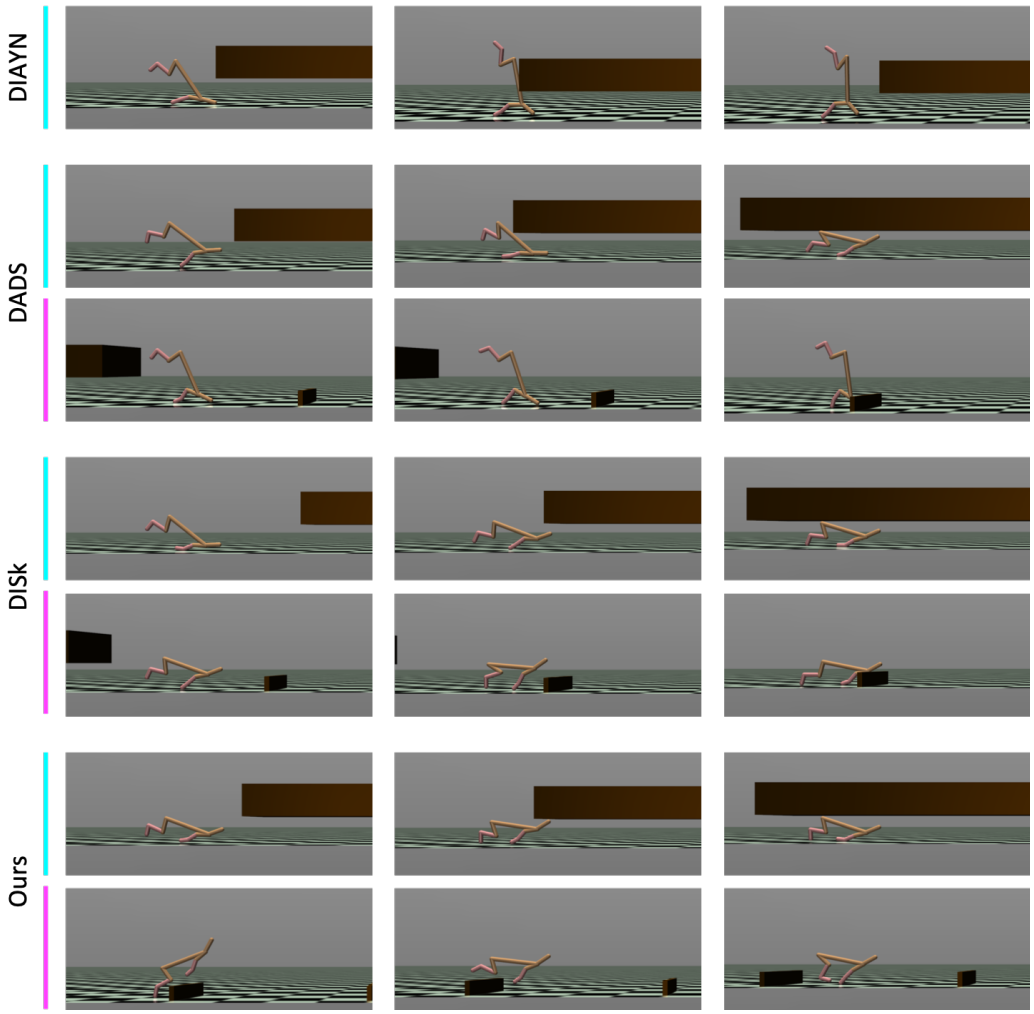

Figure 8. Visualizing skills learned for HalfCheetah-Continual-v3

D. Visualizing Learned Skills

Figure 8 shows how the learned skills handle hurdles and a tunnel in the introduced changing environment HalfCheetah-continual-v3. All algorithms except the baseline DIAYN have learned skills that allow the agent to pass through the tunnel. These skills commonly represent behaviors that make the agent lower his head or curl his front legs in front of the tunnel. Baselines cannot learn skills that enable an agent to explore the hurdle section, which is an important element of shared structures. In contrast, our algorithm learns various skills that can handle hurdles. These skills represent behaviors that make the agent lift the front and hind legs in turn or jump in front of hurdles.

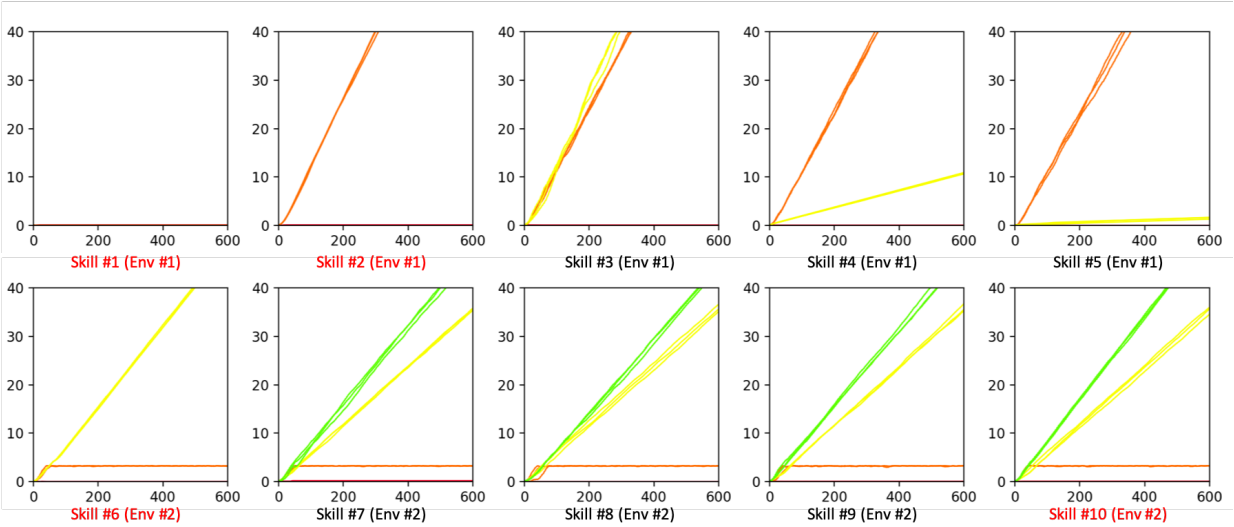

Figure 9. Visualizing skills learned with our skill evaluation process on the sequence of HalfCheetah-v3 and HalfCheetah-tunnel-v3 environments. Each plot shows the X positions of the agent over time for skills. The saved skills are marked with red text.

E. Additional Analysis of skill evaluation process

Figure 9 describes the overall skill evaluation process summarized in Figure 5 of the main paper. We discover five skills each in the first and the second obstacle configuration of HalfCheetah-continual-v3. Note that the first skill is saved without our evaluation process as we do not have skills to compare. For the first environment, our evaluation process decides to save the second skill as it is faster than the first skill. However, the other skills learned in this environment are all abandoned. It's because they represent similar or slower behaviors than the second skill.

As described in the bottom row, the skills learned in the first environment cannot handle a tunnel. In contrast, the sixth skill represents behaviors that can explore the tunnel section. Since this skill is distinct from previously acquired skills and is faster than them on the tunnel section, our evaluation process determines to save it. All subsequent skills represent faster behaviors than those of the sixth skill. However, the skills other than the last skill are not saved, as their similarities to the sixth skill are greater than the similarity threshold η.

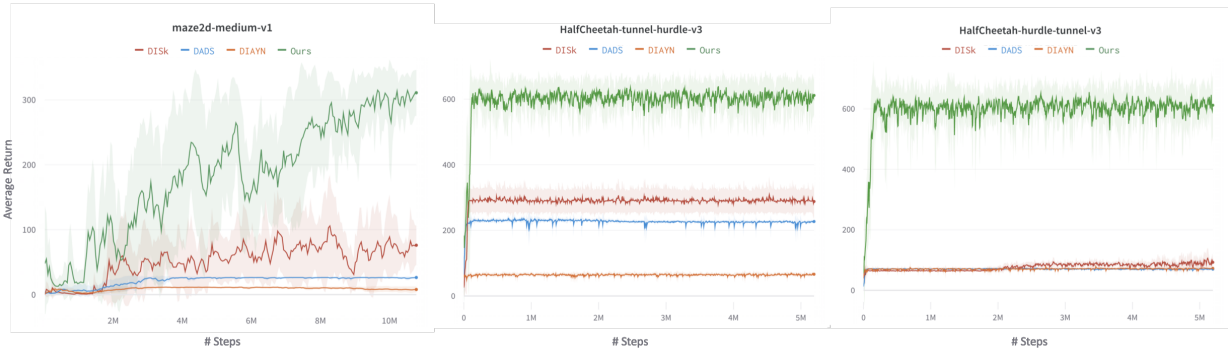

Figure 10. Learning curves for downstream tasks. The darker-colored lines and shaded areas denote the average returns and standard deviations, respectively, computed over five random seeds. **Left:** maze2d-medium-v1. **Middle:** HalfCheetah-tunnel-hurdle-sequence-v3. **Right:** HalfCheetah-hurdle-tunnel-sequence-v3.

F. Additional Results on Downstream Learning

Figure 10 describes the learning curves for the downstream tasks used in our experiments. These downstream tasks have shared structures that exist in corresponding changing environments where skills are learned. We observed that our skills accelerate learning of downstream tasks, outperforming baselines including state-of-the-art unsupervised skill discovery algorithms. This indicates that our skills adequately represent the shared structures between changing environments. Figure 11 shows the X position of the agent across time in HalfCheetah-hurdle-tunnel-sequence-v3 and HalfCheetah-tunnel-hurdle-sequence-v3, respectively.

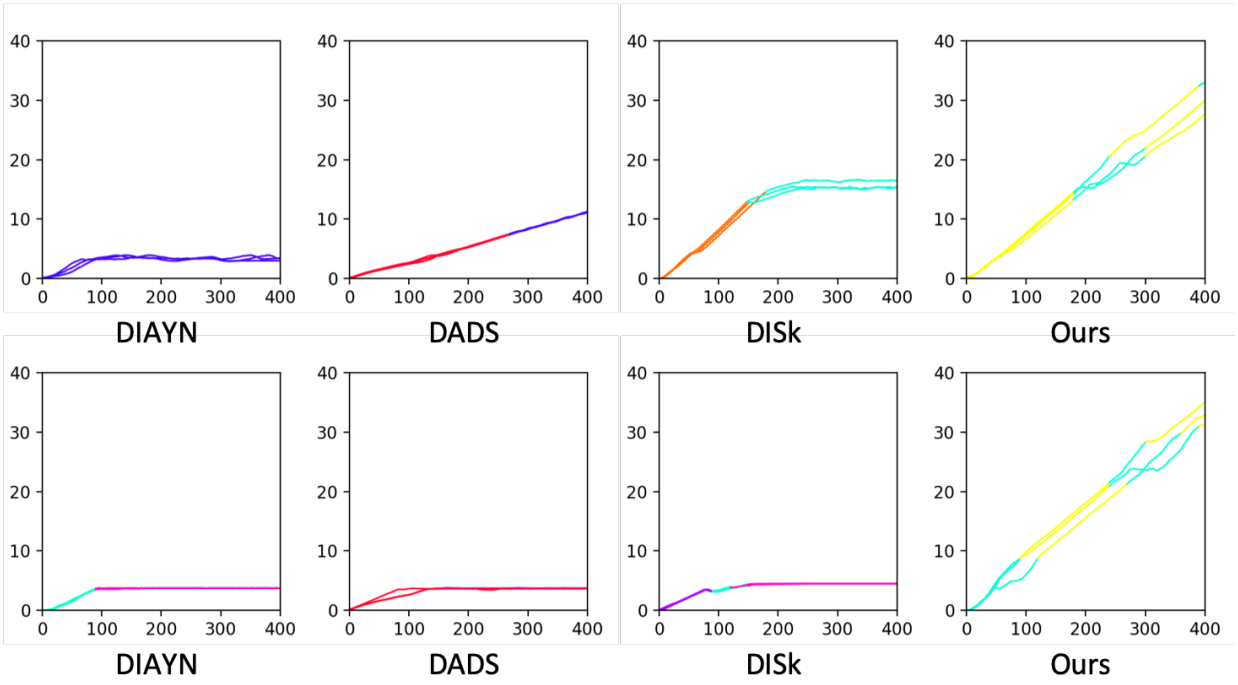

Figure 11. Trajectories of skills composed in HalfCheetah downstream tasks. Each plot shows the X position of the agent over time for composed skills. **Top:** HalfCheetah-tunnel-hurdle-sequence-v3. **Bottom:** HalfCheetah-hurdle-tunnel-sequence-v3.