Hierarchical Skills for Efficient Exploration

Anonymous Author(s)
Affiliation
Address
email

Abstract

Pre-trained, low-level skills for reinforcement learning provide higher-level action spaces with the potential to facilitate exploration. The design of such skills presents an inductive bias and is therefore subject to a trade-off between faster learning and generality across environments. In prior work on continuous control, the sensitivity of methods to this inductive bias has not been addressed explicitly, as locomotion provides a suitable prior for navigation tasks, which have been of foremost interest. In this work, we introduce a benchmark suite of sparse-reward tasks for bipedal robots demanding a variety of motor abilities. This allows us to clearly expose the consequences of trading off exploration benefits and fine-grained control. We propose a novel hierarchical skill learning framework that offloads this trade-off to high-level policy training and which produces skills that are useful across a wide range of environments. Finally, we present a three-layered hierarchical learning algorithm to perform this trade-off automatically, outperforming existing approaches to end-to-end hierarchical reinforcement learning and unsupervised skill discovery.

1 Introduction

A promising direction for improving the sample efficiency of reinforcement learning agents in complex environments is to pre-train low-level skills that are then used to structure the exploration in downstream tasks [21, 18, 25, 12, 27]. This has been studied in particular for the control of (simulated) robots, where there is a natural hierarchical decomposition of the downstream tasks into a low-level control of the robot's actuators, and a high level control signal that specifies a direction or target robot configuration with coarser temporal resolution. The large body of work on unsupervised skill or option discovery in hierarchical reinforcement learning (HRL) for continuous control relies, explicitly or implicitly, on prior knowledge that low-level skills should control the center of mass of the robot [24, 16, 12, 39, 7]. This nicely fits a wide range of benchmark tasks that are variants of navigation problems, but the benefit of such hierarchical setups outside this problem class is unclear.

The prior knowledge embedded in a pre-trained skill defines a specific trade-off between sample efficiency and generality. Skills that severely constrain the high-level action space are likely to provide the largest gains in sample efficiency, but are unlikely to be useful on a diverse set of downstream tasks. Conversely, more complex low-level skills induce a weaker inductive bias, which renders them more widely applicable but also less useful guides for exploration. There is, thus, no single universally superior pre-trained skill. Depending on the downstream task, different skills might also be useful to efficiently explore in different parts of the environment.

In this paper, we explore the feasibility of pre-training a *hierarchy of skills* of increasing complexity, that a high-level policy can compose with dynamically. In the context of simulated robots, each skill consists of controlling a part of the robot configuration on a short time horizon, such as the position of the left foot of a humanoid, or the orientation of its torso. Skills of increasing complexity are built up by jointly controlling larger portions of the configuration. These skills are pre-trained within

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

a shared policy on an environment without rewards and containing only the robot. They are then used in a downstream task within a three policies hierarchy: the higher level selects the skill (which specifies a *goal space*), the second level the target configuration within that skill (the *goal*), and finally the pre-trained skill performs the low-level control to reach the goal. Compared to standard approaches involving a single static pre-trained skill, our approach offers more flexibility to structure the exploration. It transfers the problem of choosing the prior knowledge from the pre-training phase to within the hierarchical exploration, on the other hand reducing the inductive bias, since the high-level policy needs to figure out at training time which skills are most suitable for the task.

We perform an experimental analysis of our hierarchical pre-training on a new set of challenging sparse reward tasks on bipedal robots in MuJoCo environments. Our experiments showcase that each task is more efficiently explored by its own low-level skill, confirming that even on natural tasks, where locomotion is of primal importance, there is no overall single best pre-trained skill. We further show that dynamic selection of goal spaces performs equally or better than a generic skill on all tasks, and can further improve over the best single skills per task. We achieve achieve superior results compared to existing skill discovery and HRL approaches [12, 30].

The main contributions of our work are summarized as follows:

- We propose a novel unsupervised pre-training approach that produces a hierarchy of skills based on control of variable feature sets. This allows us to off-load the trade-off between efficient exploration and fine-grained control to the application on downstream tasks.
- We demonstrate how to automatically select between different skills of varying complexity with a three-level hierarchical policy that selects skills, goals, and native actions.
- We introduce a benchmark suite of sparse-reward tasks that allows for consistent and thorough evaluation of motor skills and HRL methods beyond traditional navigation settings.
- We study the implications of inductive bias in skills experimentally, showcase the efficacy of our hierarchical skill framework on the proposed benchmark tasks, and compare their utility to existing skill discovery and HRL approaches.

2 Related Work

The success of macro-operators and abstraction in classic planning systems [13, 37] has inspired a large body of works on hierarchical approaches to reinforcement learning [8, 42, 3, 45, 30, 36]. While the decomposition of control across multiple levels of abstraction provides intuitive benefits such as easier individual learning problems and long-term decision making, recent work found that a primary benefit of HRL, in particular for modern, neural-network based learning systems, stems from improved exploration capabilities [20, 32]. From a design perspective, HRL allows for separate acquisition of low-level policies (options; skills), which can dramatically accelerate learning on downstream tasks. A variety of works propose the discovery of such low-level primitives from random walks [24, 47], mutual information objectives [16, 12, 39, 7], datasets of agent or expert traces [34, 2, 23], motion capture data [33, 28], or from dedicated pre-training tasks [14, 25].

In order to live up to their full potential, low-level skills must be useful across a large variety of downstream tasks. In practice, however, a trade-off between generality (wide applicability) and specificity (benefits for specific tasks) arises. A large portion of prior work on option discovery and HRL resolved this trade-off, explicitly or implicitly, in favor of specificity. This can be exemplified by the choice of test environments, which traditionally revolve around navigation in grid-world mazes [42, 8, 10]. In recent work on simulated robotics, navigation problems that are similar in spirit remain the benchmarks of choice [11, 30, 12, 25, 2]. In these settings, directed locomotion, i.e., translation of the robot's center of mass, is the main required motor skill, and the resulting algorithms require a corresponding inductive bias. This prior is made explicit with skills partitioning the state space according the agent's position [12, 39], or is implicitly realized by high contribution of position features to reward signals [30, 31, 2] (Appendix G). Similarly, works that target non-navigation environments acquire skills with tailored pre-training tasks [35] or in-domain motion capture data [33, 28, 29]. In contrast, our work is concerned with learning low-level skills without extra supervision from traces or pre-training task design, and which do not prescribe a fixed trade-off towards a particular type of behavior.

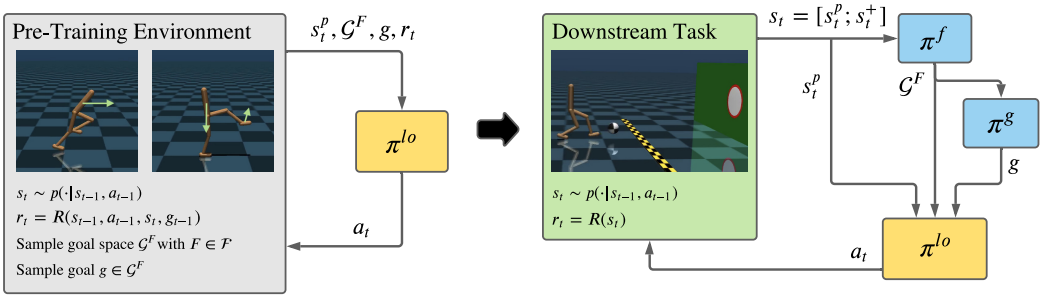


Figure 1: Illustration of our proposed hierarchical skill learning framework. **Left** Low-level policies are learned in an empty pre-training environment, with the objective to reach random configurations (goal g) of a sampled skill (goal space $\mathcal{G}^F$ defined over a feature set F). Examples of goal space features are translation along the X-axis or the position of a limb. **Right** Learning on downstream tasks with a three-level hierarchical policy to select a goal space, a goal and finally a native action a_t with the pre-trained low-level policy. The low-level policy acts on proprioceptive states s^p, while high-level policies π^{f} and π^{g} leverage extra task-specific information via s^+.

3 Hierarchical Skill Learning

In this work, we propose an approach where we first acquire low-level policies that are able to carry out a useful set of skills in an unsupervised manner, i.e., without reward signals from the main tasks of interest. We subsequently exploit it in a hierarchical reinforcement learning setting (Figure 1). The pre-training environment consists solely of the robot. In downstream tasks, additional objects may be present besides the robot; these may be fixed (e.g., obstacles) or only indirectly controllable (such as a ball).

Crucially, the usefulness of a skill policy stems from providing additional structure for effective exploration. In order to provide benefits across a wide range of tasks, skills need to support both fast, directed exploration (e.g., locomotion) as well as precise movements (e.g., lift the left foot while bending down). We propose to fulfill these requirements with short-horizon, goal-directed low-level policies that are trained to achieve target configurations of robot-level state features $\mathcal{S}^{\mathrm{g}}$, such as the position or orientation of its body or relative limb positions. This allows the definition of a hierarchy of skills by controlling single features and their combinations, resulting in varying amounts of control being exposed to high-level policies. Goal-directed skill policies are trained without task-specific rewards, relying solely on state features related to the specific robot and the prescribed hierarchy. Our low-level policies thus take the form $\left\{\pi_F^{\mathrm{lo}} : \mathcal{S}^{\mathrm{p}} \times \mathcal{G}^F \to A\right\}_{F \in \mathcal{F}}$ with proprioceptive observations $\mathcal{S}^{\mathrm{p}}$ and goal spaces $\mathcal{G}^F \subseteq \mathcal{S}^{\mathrm{g}}$ defined over feature sets $\mathcal{F}$.

On downstream tasks, high-level policies operate with a combinatorial action space as the skill policy π^{lo} is conditioned on both a feature set F and a concrete goal $g \in \mathcal{G}^F$. State spaces are enriched with task-specific features $\mathcal{S}^+$, s.t. $\mathcal{S} = \mathcal{S}^{\mathrm{p}} \cup \mathcal{S}^+$, that contain information on the additional objects in the downstream task. This enriched state space is only available to the high-level policy. We model the high-level policy in a hierarchical fashion with two policies $\pi^{\mathrm{f}} : S \to \mathcal{F}$ and $\pi^{\mathrm{g}} : S \times \mathcal{F} \to \mathcal{G}^F$ that prescribe a goal space and a goal, respectively. With a task- and state-dependent policy π^{f}, it is possible to not only select the required set of skills for a given environment, but also to switch between them during episodes. In this three-level hierarchy, the higher-level policy explores the hierarchy of goal spaces and dynamically trades off between generality and specificity. Temporal abstraction is obtained by selecting new high-level actions at regular intervals but with a reduced frequency compared to low-level actions. We term the resulting learning algorithm *HSD-3*, emphasizing the three different levels of control (goal spaces, goals and native actions) obtained after our hierarchical skill discovery phase.

3.1 Unsupervised Pre-Training

During pre-training, skill policies act in an MDP $\mathcal{M}$ with proprioceptive states s^p, native actions a, transition probabilities and initial states $s_0^p \in \mathcal{S}_0^p$ drawn from a distribution P_0 [41, for example], but without an extrinsic reward function. The skill policies are trained to reach goals in a goal space S^g consisting of robot-level sensor readings, which may include non-proprioceptive information.

Critically, each skill π^{lo}_F aims to achieve goals defined over a variable-sized subset of features F of S^g. To avoid learning a combinatorial number of independent policies, we follow the principle of universal value function approximators [38] and share the parameters of each skill. We augment the policy's input with goal and goal space information, and learn a single set of parameters θ, leading to $\pi^{lo}_\theta : \mathcal{S}^{\text{p}} \times \mathcal{F} \times \mathcal{G} \rightarrow \mathcal{A}$, where $\mathcal{F} := \{F : F \subseteq \mathcal{S}^{\text{g}}\}$ and $\mathcal{G} := \{\mathcal{G}^F : F \subseteq \mathcal{S}^{\text{g}}\}$. For ease of modeling, F is provided to π^{lo}_θ as a bag-of-words input where a coordinate is set to 1 if its respective feature is included in F and to 0 otherwise. Also, the low-level policy receives goals relative to the current values of the s^{g}. Thus, at each time step, the input goal to the policy is updated to reflect the progress towards the goal. As a learning signal, we provide a distance-based reward as $R(s^g, a, s^{g'}, F, g) := ||\omega^F(s^g) - g||_2 - ||\omega^F(s^{g'}) - g||_2$, where $\omega^F : \mathcal{S}^{\text{g}} \rightarrow \mathcal{G}^F$ is a fixed transformation that selects the subset F of goal space features and applies a suitable normalization. The selection of features for $\mathcal{S}^{\text{g}}$ and transformations ω^F represents the inductive bias utilized during unsupervised pre-training. We discuss examples for bipedal robots in our experiments and in Appendix B.

During training, features F and goals g are sampled anew for each episode in the environment. Episodes consist of several dozen steps only, reflecting the fact that we are interested in short-horizon skills. In-between episodes, we reset the simulation to states drawn from $\mathcal{S}^p_0$ only sporadically to encourage the resulting policies to be applicable in a wide range of states, facilitating their sequencing under variable temporal abstractions. Likewise, rewards are propagated across episodes as long as the simulation is not reset. The low-level policy parameters θ can be optimized with any reinforcement learning algorithm; here, we opt for Soft Actor-Critic (SAC), a state-of-the-art method tailored to continuous action spaces [17]. Pseudo-code for the pre-training algorithm and further implementation details are provided in Appendix C.

3.2 Hierarchical Control

After a low-level policy π^{lo} has been obtained via unsupervised pre-training, we employ it to construct a full hierarchical policy $\pi(a|s) = \pi^{\text{f}}(F|s)\pi^{\text{g}}(g|s, F)\pi^{\text{lo}}(a|s^p, F, g)$. The hierarchical policy is applied to downstream tasks, where we are supplied with a reward signal and state observations $\mathcal{S} = \mathcal{S}^{\text{p}} \cup \mathcal{S}^+$. While different frequencies may be considered for acting with π^{f} and π^{g}, in this work both high-level actions are selected in lock-step. For jointly learning π^{f} and π^{g}, we extend the SAC algorithm to incorporate the factored action space $\mathcal{F} \times \mathcal{G}$. In particular, it is necessary to extend SAC to deal with a joint discrete and continuous action space, which we describe below (see Appendix D for more details).

We formulate our extension with a shared critic $Q(s, F, g)$, incorporating both factors of the action space. $Q(s, F, g)$ is trained to minimize the soft Bellman residual [17, Eq. 5], with the soft value function $V(s)$ including entropy terms for both policies. Since π^{f} outputs a distribution over discrete actions (to select from available goal spaces), we compute its entropy in closed form instead of using the log-likelihood of the sampled action as for π^{g}. We employ separate temperatures α and β^F for the goal space policy as well as for each corresponding goal policy; β^F are normalized to account for varying action dimensions across goal spaces.

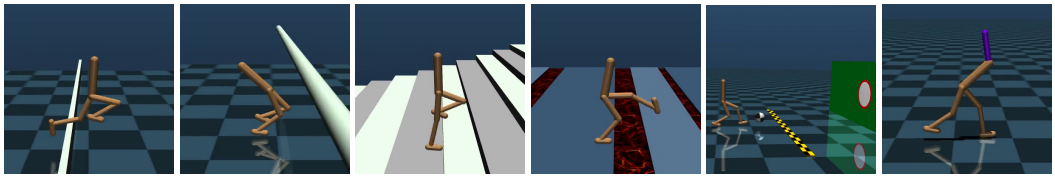
$$V(s) = \sum_{F \in \mathcal{F}} \pi^{\text{f}}(F|s) \mathop{\mathbb{E}}_{g \sim \pi^{\text{g}}} \left[Q(s, F, g) - \frac{\beta^F}{|F|} \log \pi^{\text{g}}(g|s, F) \right] + \alpha \left(\mathcal{H}(\pi^f(\cdot|s)) - \log |\mathcal{F}| \right).$$

Compared to the usual SAC objective for continuous actions, we subtract $\log |\mathcal{F}|$ from the entropy to ensure a negative contribution to the reward. This matches the sign of the entropy penalty from π^{g}, which is negative in standard hyper-parameter settings[1] [17]. Otherwise, the entropy terms for discrete and continuous actions may cancel each other in Q-function targets computed from $V(s)$.

The policy loss [17, Eq. 7] is likewise modified to include both entropy terms. Parameterizing the policies with ϕ and ψ and the Q-function with ρ, we obtain

$$J_{\pi^{f,g}}(\phi, \psi) = \mathop{\mathbb{E}}_{s \sim B} \left[\sum_{F \in \mathcal{F}} \pi^{\text{f}}_\phi(F|s) \mathop{\mathbb{E}}_{g \sim \pi^{\text{g}}_\psi} \left[\frac{\beta^F}{|F|} \log \pi^{\text{g}}_\psi(g|F, s) - Q_\rho(s, F, g) \right] - \alpha \, \mathcal{H}(\pi^{\text{f}}_\phi(\cdot|s)) \right],$$

[1] The entropy for continuous actions sampled from a Gaussian distribution can be negative since it is based on probability densities.



(a) Hurdles (b) Limbo (c) Stairs (d) Gaps (e) GoalWall (f) PoleBalance

Figure 2: Benchmark environments for evaluating motor skills for bipedal robots, pictured with the Walker robot. Hurdle and Limbo bar heights and spacing, as well as stair lengths, are sampled randomly from fixed distributions. We also include a combination of Hurdles and Limbo (HurdesLimbo), in which both obstacle types alternate.

where the replay buffer is denoted with B. The scalar-valued entropy temperatures are updated automatically during learning to maintain target entropies $\overline{\mathcal{H}}^f$ and $\overline{\mathcal{H}}^g$ for both π^{f} and π^{g}. The following losses are minimized at each training step:

$$J(\alpha) = \mathop{\mathbb{E}}_{s\sim B}\left[\alpha\left(\mathcal{H}(\pi^{\mathrm{f}}(\cdot|s)) - \overline{\mathcal{H}}^{\Omega}\right)\right]$$

$$J(\beta) = \mathop{\mathbb{E}}_{s\sim B}\left[-\sum_{F\in\mathcal{F}}\beta^F\pi^{\mathrm{f}}(F|s)\mathop{\mathbb{E}}_{g\sim\pi^{\mathrm{g}}}\left[\log\pi^{\mathrm{f}}(g|s,F) + \overline{\mathcal{H}}^g\right]\right].$$

We implement temporal abstraction by selecting high-level actions with a reduced frequency. This incurs a reduction in available training data since, when taking a high-level action every c steps in an N-step episode, we obtain only N/c high-level transitions. To leverage all available transitions gathered during training, we adopt the step-conditioned critic proposed by Whitney et al. [47] in our SAC formulation. The Q-function receives an additional input $0 \leq i \leq c$ marking the number of steps from the last high-level action, and is trained to minimize a modified soft Bellman residual:

$$J_Q(\rho) = \mathop{\mathbb{E}}_{\substack{F_t, g_t, i,\\ s_{t,\ldots,t+c-i},\\ a_{t,\ldots,t+c-i}\sim B}}\left[\frac{1}{2}\left(Q_\rho(s_t, F_t, g_t, i) - \left(\sum_{j=0}^{c-i-1}\left(\gamma^j r(s_{t+j}, a_{t+j})\right) + \gamma^{c-i}V(s_{t+c-i})\right)\right)^2\right]$$

As in [47], $Q(s, F, g, 0)$ is used when computing $V(s)$ and the policy loss $J_{\pi^{f,g}}(\phi, \psi)$.

The factorized high-level action space of HSD-3 has previously been studied as parameterized action spaces [26, 46], albeit with a small number of discrete actions and not in the context of hierarchical RL. A possible application of Soft Actor-Critic is described in Delalleau et al. [9]; our approach differs in that we (a) compute the soft value function as a weighted sum over all discrete actions, and (b) opt for two separately parameterized actors and a shared critic. These design choices proved to be more effective in initial experiments.

4 Benchmark Environments

We propose a benchmark suite for comprehensive evaluation of pre-trained motor skills, tailored to bipedal robots (Figure 2) and implemented for the MuJoCo physics simulator [44, 43]. We selected tasks that require diverse abilities such as jumping (Hurdles), torso control (Limbo), fine-grained foot control (Stairs, GoalWall) and body balance (PoleBalance). Besides PoleBalance, all tasks also require locomotion. The tasks are designed to have a sparse reward that is provided once an obstacle or step has been passed or a goal has been shot, emphasizing the need for good exploration. As an exception, PoleBalance provides a constant reward during the entire episode, which ends when the pole placed on the robot falls over. PoleBalance requires precise motor control, and is, in combination with its reward structure, chiefly outside of the realm of benchmarks that have been traditionally used to evaluate HRL techniques. Obstacle position and step lengths are subject to random perturbations, sampled anew for each episode. We tailor several environment parameters to the concrete robot being controlled, e.g., the positions of Limbo bars. For the effectively two-dimensional Walker robot, we limit the movement of objects (ball, pole) to translation along X and Z and rotation around the Y-axis.

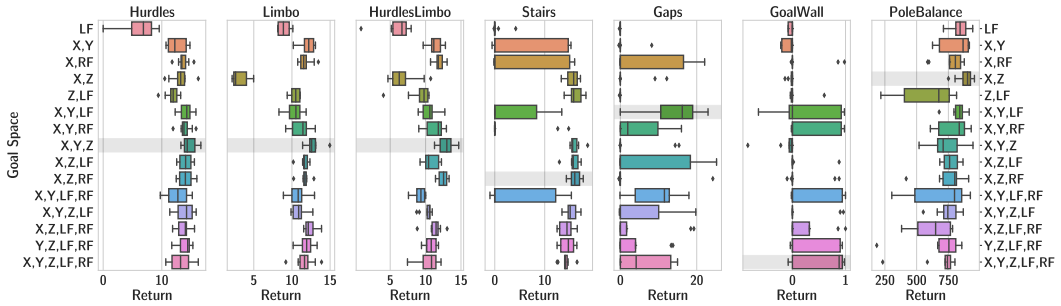


Figure 3: Returns achieved after 5M samples on the benchmark tasks with the Walker robot with fixed low-level policy goal spaces (quartiles and means). Each row corresponds to a set of features for the respective goal space. Best skills (marked) differ significantly across tasks.

In all environments, we label simulation states that would cause the robot to fall over as invalid; these will terminate the episode with a reward of -1. Appendix A contains full specifications for all tasks.

Variants of some of our benchmark tasks have been proposed in prior works: Hurdles [19, 22, 12, 35], Stairs [15], Gaps [19, 43], and PoleBalance is inspired by the classical CartPole problem [5]. Our tasks are different from earlier published counterparts in parameterization and in the sparsity of rewards they provide.

5 Experimental Results

For our experiments, we first pre-train skill policies as described in Section 3.1 in an empty environment. The feature set to describe target configurations consists of the robot's translation along the X-axis, its position on the Z-axis, torso rotation around the Y-axis and positions of each foot, relative to the body (Appendix B). With 5 individual features, we obtain a set of $2^5 - 1 = 31$ skills with a shared set of policy parameters. We then train a separate high-level policy for each benchmark environment, directing the pre-trained skills. High-level actions are taken every 5 environment steps; in the PoleBalance environment, which requires short reaction times, all policies operate at the same frequency as the environment. All neural network policies employ skip connections as recently proposed for SAC by Sinha et al. [40]. Pre-training takes approximately 3 days on 2 GPUs (V100) and fine-tuning (on task training) takes 2 days to reach 5M samples on 1 GPU. Further details regarding the training setup, hyper-parameters for skill and high-level policy training, as well as for baselines, can be found in Appendix E.

5.1 Inductive Bias of Skills

In a first experiment with the Walker robot, we aim to investigate the trade-off between specificity and generality of low-level skills. For each environment, we train high-level policies directing a *fixed* skill within its respective goal space, which for example consists of torso-related features only, or of feet positions combined with forward translation. At the end of training, we measure average returns obtained for all 31 skills in a deterministic evaluation setting, and plot means and quartiles over different seeds in Figure 3. The performance for different fixed skills varies significantly across environments, and *no single skill is able to obtain the best result (shaded) across all environments*. While controlling X, Y and Z features only produces the best hierarchical policies for the Hurdles, Limbo, and HurdlesLimbo tasks, these yield poor results for the Gaps, GoalWall and PoleBalance tasks. Controlling feet positions (LF and RF) is crucial for GoalWall, although good performance is only obtained for few runs. The most general option, high-level control of *all* considered features (bottom line), allows for learning progress on all tasks; however, with the exception of GoalWall, it is outperformed by other, more specific skills.

5.2 Skill Selection with HSD-3

We compare HSD-3 against the following baselines to examine the effectiveness of our proposed framework for skill learning and hierarchical control:

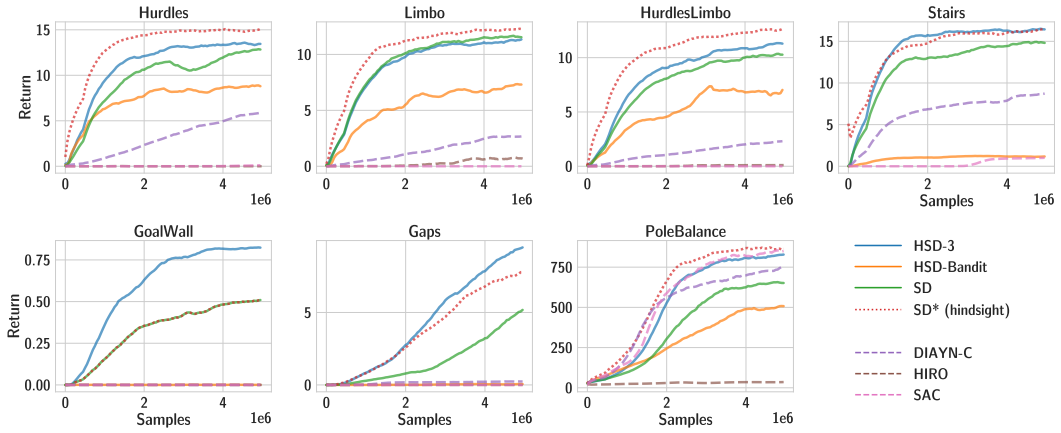


Figure 4: Learning Curves on benchmark environments with the Walker robot. For clarity, we plot mean performance over 9 seeds, averaged over 0.5M environment steps. Full learning curves including error bands are provided in Appendix F.

- **Non-Hierarchical RL:** Soft Actor-Critic (SAC) baseline, which is also used for training all our hierarchical settings [17];
- **End-to-End Hierarchical RL:** Instead of pre-training skills, we attempt to learn them in an end-to-end fashion with SAC. Our HRL algorithm closely resembles HIRO [30], a state-of-the-art HRL method for continuous control, and we implement a similar goal relabeling strategy to improve data efficiency. The goal space for directing the low-level policy consists of the full feature set used for our hierarchical skill pre-training.
- **Unsupervised Skill Discovery** We evaluate DIAYN as a representative unsupervised pre-training objective [12]. For fair comparison, we provide the skill discriminator with the same set of goal features that are used in our pre-training stage. We obtain a continuous-valued high-level action space by embedding DIAYN's discrete skill variable into a continuous space as proposed by Achiam et al. [1], and hence denote this baseline with *DIAYN-C*.
- **Full Goal Space** We may opt to not learn a high-level policy for goal space selection and use a fixed set of features F instead. Without sufficient prior knowledge about the task at hand, this corresponds to the full goal space $F = \{0, 1, ..., \dim(\mathcal{S}^g)\}$, settling for maximum generality. We refer to this baseline as *SD*. We further compare to the the *best per-task* goal spaces from Section 5.1, denoted with *SD**, which we regard as a topline as it requires exhaustive training and evaluation with all available goal spaces and selection in hindsight.
- **Per-Task Goal Space Selection** For a given task, a suitable single goal space $\mathcal{G}^F$ can also be derived from experience gathered during training. For exploration, we utilize the sliding-window UCB bandit algorithm proposed in Badia et al. [4]. Each arm corresponds to a candidate goal space that will be selected for an entire training episode, and the reward is provided by the total return collected over this episode.

Method	Hurdles	Limbo	HurdlesLimbo	Stairs	Gaps	GoalWall	PoleBalance
SAC	0.1±0.4	0.0±0.0	-0.1±0.1	1.1±3.2	-0.1±0.1	0.0±0.0	854.7±96.5
HIRO-SAC	0.0±0.0	0.5±1.7	0.1±0.3	-	0.0±0.0	-0.1±0.1	38.3±11.8
DIAYN-C	6.1±3.2	2.8±2.4	2.3±1.7	8.8±2.3	0.2±0.4	-0.1±0.3	794.9±178.7
SD	12.5±1.4	11.3±0.5	10.4±0.6	14.7±1.2	5.8±7.0	0.5±0.5	672.6±177.0
HSD-Bandit	8.6±1.6	7.1±2.4	9.8±2.1	1.2±0.2	0.0±0.1	0.0±0.1	493.1±173.2
HSD-3	13.6±0.5	11.5±1.1	10.8±1.5	16.4±1.1	9.8±8.7	0.8±0.3	847.1±62.4
SD*	15.3±1.1	12.3±1.4	13.5±0.7	16.5±1.2	8.2±9.4	0.5±0.5	887.7±68.5

Table 1: Final performance after 5M samples across benchmark tasks with the Walker robot.

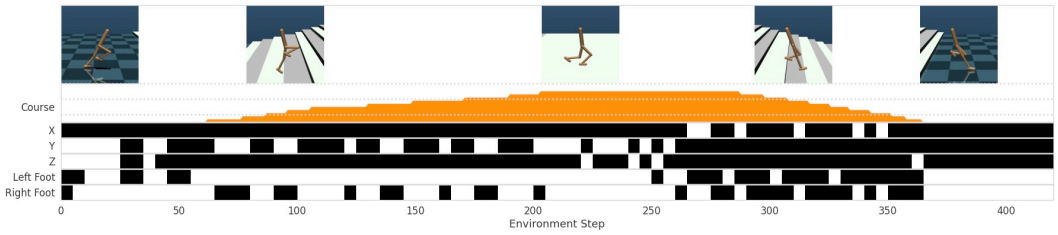


Figure 5: Skills selected by HSD-3 during an episode on the Stairs task. Different skills are utilized for different sections of the environment: walking upstairs requires control of a foot by the high-level policy, and walking downstairs requires more careful treading with both feet being controlled. On straight sections (start, middle, end), fast locomotion is achieved by jointly controlling the torso position (X, Z) and orientation (Y).

In Figure 4, we plot learning curves over 5M training interactions for HSD-3 as well as base- and toplines, and list final performance in Table 1. On all tasks, HSD-3 outperforms or matches (Limbo) the learning speed of a conventional hierarchical policy with a skill defined over the full goal space (SD). On GoalWall and Gaps, the *three-level policy significantly exceeds the mean performance of single goal space base- and toplines*. For these two tasks, the variances reported in Table 1 are high, hinting at especially challenging exploration problems. A closer look at per-seed performance reveals that for GoalWall, 8 out of 9 HSD-3 training runs obtain returns above 0.8; for SD, this applies to 5 runs only. On the Gaps task, the average return among succeeding runs is 17.6 for HSD-3 and 10.5 for SD. Goal space selection with a bandit algorithm (HSD-Bandit) reduces learning speed compared to other methods for utilizing our pre-trained skills, and fails on the GoalWall, Gaps and Stairs tasks. DIAYN-C exhibits with mediocre performance across tasks, with the exception of PoleBalance where learning is fast and returns are higher than for SG. The end-to-end HRL method HIRO-SAC is not able to make meaningful progress on any task, which highlights the utility of skill pre-training in the absence of task-specific priors. Standard SAC excels in PoleBalance, which has a dense reward structure compared to the remaining environments. Conversely, no meaningful learning progress is achieved in sparse-reward settings.

These results demonstrate that, without additional task-specific knowledge, our hierarchical skill framework is able to *automatically and efficiently trade off between skills of varying complexity at training time*. In Figure 5, we visualize the goal spaces that are selected across the course of an episode in the Stairs environment. During different stages of the environment (walking upstairs, downstairs, and on a planar surface), HSD-3 selects appropriate skills of varying complexity.

5.3 Evaluation on a Humanoid Robot

To evaluate the scalability of HSD-3 to more complex robots, we perform an additional set of experiments with the challenging 21-joint Humanoid robot from [43]. For skill pre-training, we select a goal space matching the Walker setup, consisting of the translation along the X-axis of simulation, Z position of the torso, its rotation around the Y axis and three-dimensional feet positions relative to the robot's hip location.

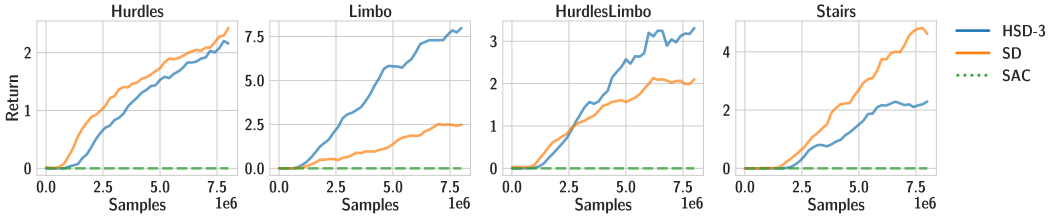


Figure 6: Results on four of the benchmark tasks with a 21-joint Humanoid robot. We show mean performance over 3 seeds, averaged over 0.5M environment steps. Negative return are now show, return is clipped at 0. Appendix F contains full learning curves including error bands.

In Figure 6, we plot average learning progress across 8M environment steps, comparing HSD-3 to SD and a standard, non-hierarchical SAC setup. Both HSD-3 and SD make good learning progress on the three out of four tasks used for this evaluation. In Limbo, HSD-3 is better able to learn faster by selecting a pre-trained skill to control forward motion and body height. We found that this behavior was consistently learned in all runs, but was not discovered by the SD baseline, which exercises control over all goal space features. On the Stairs task, however, the baseline utilizing the full goal space exhibits better performance. In this preliminary study, none of the hierarchical policies was able to make significant progress on the GoalWall and Gaps tasks, which were already shown to be difficult with the Walker robot in Section 5.2.

6 Conclusion

Our work is among the firsts to highlight the specific trade-offs stemming from the inductive bias provided by pre-trained skills, and to demonstrate its consequences experimentally in simulated robotic control settings. We describe a hierarchical skill learning framework that, compared to existing approaches, allows high-level policies to learn to perform the trade-off between efficient exploration (bigger skills) and fine-grained control automatically during training on downstream tasks. We further experiment with a challenging Humanoid robot and obtain promising results when utilizing pre-trained skills over a restricted set of goal space features.

With this work, we provide a new set of benchmark tasks that require diverse motor skills and pose exploration challenges. We release these tasks in a dedicated distribution, with the aim of spurring further research on motor skills in continuous control settings, and to ultimately broaden the utility of hierarchical approaches to control to tasks beyond navigation.

Limitations: Our investigations with the Humanoid robot have been performed in limited goal spaces and hence with significant priors. We believe that further work is required to successfully acquire a large, exhaustive set of low-level skills in unsupervised environments that work on more and more complex morphologies.

References

[1] J. Achiam, H. Edwards, D. Amodei, and P. Abbeel. Variational Option Discovery Algorithms. *arXiv:1807.10299 [cs]*, Jul 2018.

[2] A. Ajay, A. Kumar, P. Agrawal, S. Levine, and O. Nachum. OPAL: Offline Primitive Discovery for Accelerating Offline Reinforcement Learning. In *International Conference on Learning Representations*, 2021.

[3] P.-L. Bacon, J. Harb, and D. Precup. The Option-Critic Architecture. In *Thirty-First AAAI Conference on Artificial Intelligence*, pages 1726–1734, 2017.

[4] A. P. Badia, B. Piot, S. Kapturowski, P. Sprechmann, A. Vitvitskyi, D. Guo, and C. Blundell. Agent57: Outperforming the Atari Human Benchmark. In *International Conference on Machine Learning*, pages 507–517, 2020.

[5] A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5):834–846, 1983.

[6] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI Gym. *arXiv:1606.01540 [cs]*, Jun 2016.

[7] V. Campos, A. Trott, C. Xiong, R. Socher, X. Giro-i-Nieto, and J. Torres. Explore, Discover and Learn: Unsupervised Discovery of State-Covering Skills. In *International Conference on Machine Learning*, pages 1317–1327, 2020.

[8] P. Dayan and G. E. Hinton. Feudal Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 5, pages 271–278. Morgan-Kaufmann, 1992.

[9] O. Delalleau, M. Peter, E. Alonso, and A. Logut. Discrete and Continuous Action Representation for Practical RL in Video Games. In *AAAI-20 Workshop on Reinforcement Learning in Games*, 2019.

[10] T. G. Dietterich. Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

[11] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel. Benchmarking Deep Reinforcement Learning for Continuous Control. In *International Conference on Machine Learning*, pages 1329–1338, 2016.

[12] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is All You Need: Learning Skills without a Reward Function. In *International Conference on Learning Representations*, 2019.

[13] R. E. Fikes, P. E. Hart, and N. J. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 3:251–288, 1972.

[14] C. Florensa, Y. Duan, and P. Abbeel. Stochastic Neural Networks for Hierarchical Reinforcement Learning. In *International Conference on Learning Representations*, 2017.

[15] D. Ghosh, A. Singh, A. Rajeswaran, V. Kumar, and S. Levine. Divide-and-Conquer Reinforcement Learning. In *International Conference on Learning Representations*, 2018.

[16] K. Gregor, D. J. Rezende, and D. Wierstra. Variational Intrinsic Control. In *International Conference on Learning Representations (Workshop)*, 2017.

[17] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine. Soft Actor-Critic Algorithms and Applications. *arXiv:1812.05905v2 [cs]*, Dec 2018.

[18] K. Hausman, J. T. Springenberg, Z. Wang, N. Heess, and M. Riedmiller. Learning an Embedding Space for Transferable Robot Skills. In *International Conference on Learning Representations*, 2018.

[19] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. M. A. Eslami, M. Riedmiller, and D. Silver. Emergence of Locomotion Behaviours in Rich Environments. *arXiv:1707.02286 [cs]*, Jul 2017.

[20] N. K. Jong. The utility of temporal abstraction in reinforcement learning. In *7th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 299–306, 2008.

[21] G. Konidaris. Building Portable Options: Skill Transfer in Reinforcement Learning. In *IJCAI*, volume 7, pages 895–900, 2007.

[22] A. Li, C. Florensa, I. Clavera, and P. Abbeel. Sub-policy Adaptation for Hierarchical Reinforcement Learning. In *International Conference on Learning Representations*, 2019.

[23] T. Li, R. Calandra, D. Pathak, Y. Tian, F. Meier, and A. Rai. Planning in Learned Latent Action Spaces for Generalizable Legged Locomotion. *arXiv:2008.11867 [undefined]*, 6(2):2682–2689, Mar 2021.

[24] M. C. Machado, M. G. Bellemare, and M. Bowling. A Laplacian Framework for Option Discovery in Reinforcement Learning. In *International Conference on Machine Learning*, 2017.

[25] K. Marino, A. Gupta, R. Fergus, and A. Szlam. Hierarchical RL Using an Ensemble of Proprioceptive Periodic Policies. In *International Conference on Learning Representations*, 2019.

[26] W. Masson, P. Ranchod, and G. Konidaris. Reinforcement Learning with Parameterized Actions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), 2016.

[27] J. Merel, A. Ahuja, V. Pham, S. Tunyasuvunakool, S. Liu, D. Tirumala, N. Heess, and G. Wayne. Hierarchical Visuomotor Control of Humanoids. In *International Conference on Learning Representations*, 2019.

[28] J. Merel, L. Hasenclever, A. Galashov, A. Ahuja, V. Pham, G. Wayne, Y. W. Teh, and N. Heess. Neural Probabilistic Motor Primitives for Humanoid Control. In *International Conference on Learning Representations*, 2019.

[29] J. Merel, S. Tunyasuvunakool, A. Ahuja, Y. Tassa, L. Hasenclever, V. Pham, T. Erez, G. Wayne, and N. Heess. Catch & Carry: Reusable Neural Controllers for Vision-Guided Whole-Body Tasks. *ACM Transactions on Graphics*, 39(4), 2020.

[30] O. Nachum, S. S. Gu, H. Lee, and S. Levine. Data-Efficient Hierarchical Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 31, pages 3303–3313. Curran Associates, Inc., 2018.

[31] O. Nachum, S. Gu, H. Lee, and S. Levine. Near-Optimal Representation Learning for Hierarchical Reinforcement Learning. In *International Conference on Learning Representations*, 2019.

[32] O. Nachum, H. Tang, X. Lu, S. Gu, H. Lee, and S. Levine. Why Does Hierarchy (Sometimes) Work So Well in Reinforcement Learning? In *NeurIPS 2019 Deep Reinforcement Learning Workshop*, 2019.

[33] X. B. Peng, M. Chang, G. Zhang, P. Abbeel, and S. Levine. MCP: Learning Composable Hierarchical Control with Multiplicative Compositional Policies. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[34] K. Pertsch, Y. Lee, and J. J. Lim. Accelerating Reinforcement Learning with Learned Skill Priors. In *4th Conference on Robot Learning (CoRL)*, 2020.

[35] A. H. Qureshi, J. J. Johnson, Y. Qin, T. Henderson, B. Boots, and M. C. Yip. Composing Task-Agnostic Policies with Deep Reinforcement Learning. In *International Conference on Learning Representations*, 2020.

[36] M. Riedmiller, R. Hafner, T. Lampe, M. Neunert, J. Degrave, T. Wiele, V. Mnih, N. Heess, and J. T. Springenberg. Learning by Playing - Solving Sparse Reward Tasks from Scratch. In *International Conference on Machine Learning*, pages 4344–4353. PMLR, 2018.

[37] E. D. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5(2): 115–135, 1974.

[38] T. Schaul, D. Horgan, K. Gregor, and D. Silver. Universal Value Function Approximators. In *International Conference on Machine Learning*, pages 1312–1320, 2015.

[39] A. Sharma, S. Gu, S. Levine, V. Kumar, and K. Hausman. Dynamics-Aware Unsupervised Discovery of Skills. In *International Conference on Learning Representations*, 2020.

[40] S. Sinha, H. Bharadhwaj, A. Srinivas, and A. Garg. D2RL: Deep Dense Architectures in Reinforcement Learning. In *NeurIPS 2020 Deep RL Workshop*, 2020.

[41] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning Series. The MIT Press, Cambridge, Massachusetts, second edition edition, 2018. ISBN 978-0-262-03924-6.

[42] R. S. Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999.

[43] Y. Tassa, S. Tunyasuvunakool, A. Muldal, Y. Doron, P. Trochim, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, and N. Heess. Dm_control: Software and Tasks for Continuous Control. *arXiv:2006.12983 [undefined]*, 6:100022, Nov 2020.

[44] E. Todorov, T. Erez, and Y. Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012.

[45] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu. FeUdal Networks for Hierarchical Reinforcement Learning. In *International Conference on Machine Learning*, pages 3540–3549, 2017.

[46] E. Wei, D. Wicke, and S. Luke. Hierarchical Approaches for Reinforcement Learning in Parameterized Action Space. In *2018 AAAI Spring Symposia*. AAAI Press, Stanford University, Palo Alto, California, USA, 2018.

[47] W. Whitney, R. Agarwal, K. Cho, and A. Gupta. Dynamics-aware Embeddings. In *International Conference on Learning Representations*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Details in Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We include code and instructions to reproduce the main results in the supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] . This information is provided in the main text or in the Appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We report error bars for all results where they are relevant, e.g. Figures 3, 8, and Appendix F.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We specify all compute details necessary to reproduce our results in Section 5 and Appendix E.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] (Not really a dataset but an environment, that we cite [44, 43].)
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We provide our code in the supplemental material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]