MRS: Multi-Resolution Subgoals for HRL Agents

Anonymous authors
Paper under double-blind review

Abstract

Hierarchical reinforcement learning (HRL) decomposes the policy into a manager and a worker, which enables long-horizon planning but introduces a performance gap on tasks requiring agility. We identify one of the root causes in subgoal-based HRL: the manager's goal representation is typically learned without constraints on reachability or temporal distance from the current state, preventing precise local subgoal selection. We further show that the optimal subgoal distance is both task- and state-dependent: nearby subgoals enable precise control but amplify prediction noise, while distant subgoals produce smoother motion at the cost of geometric precision. We propose Multi-Resolution Subgoals (MRS), which learns multiple goal-prediction modules, each specialized to a fixed temporal horizon, with a jointly trained meta-controller that selects among them based on the current state. MRS consistently outperforms fixed-resolution baselines and reduces the performance gap between HRL and non-HRL state-of-the-art on DeepMind Control Suite, Gym-Robotics, and long-horizon AntMaze tasks. [Project page: `https://sites.google.com/view/multi-res-skills/home`]

1 Introduction

Hierarchical reinforcement learning (HRL) decomposes the policy into a manager that proposes temporally abstract subgoals and a worker that executes primitive actions to reach them (Sutton et al., 1999; Dayan & Hinton, 1992; Vezhnevets et al., 2017). This decomposition enables effective long-horizon planning by reducing the effective decision horizon for each component (Hafner et al., 2022; Nachum et al., 2018b). However, it also introduces a structural information bottleneck at the manager-worker interface: the manager must compress its intent into a finite-dimensional goal signal, and the worker is conditioned on that compressed representation. While this bottleneck is acceptable for long-horizon tasks where coarse guidance suffices, it limits the precision available to the manager for agility-demanding tasks such as dense-reward locomotion and manipulation benchmarks, where precise, reactive control is essential. As a result, HRL agents consistently underperform monolithic agents on such tasks (Hafner et al., 2022; 2023), hindering practical adoption in settings that require both long-horizon competence and fine-grained control.

One source of this gap, in subgoal-based HRL, is the structure of the learned goal representation. The manager proposes subgoals from a goal space that is typically learned without constraints on reachability or temporal distance from the current state. Thus, the manager can, in principle, propose any state as a subgoal, regardless of whether the worker can reach it (Nachum et al., 2018b; Hafner et al., 2022; Vezhnevets et al., 2017). This unconstrained goal space has two consequences: the worker is frequently directed toward unreachable targets (Zhang et al., 2020), and representational capacity is diluted across the full state manifold rather than concentrated on the locally reachable states relevant for precise control (Nachum et al., 2018a). Together, these compound the information bottleneck and contribute to the precision deficit observed in practice.

A related but distinct difficulty concerns not the structure of the goal space but the distance of the proposed subgoal. Even if the goal space were appropriately constrained, the optimal subgoal distance is neither fixed nor task-invariant. We illustrate this with a simulation of a point agent navigating paths with straight segments and sharp turns (Fig. 1). When the subgoal is nearby, the agent follows curves precisely but is sensitive to prediction noise, where small errors produce large angular deviations. When the subgoal is distant, the agent moves smoothly but accumulates geometric error at turns. Neither strategy dominates across all

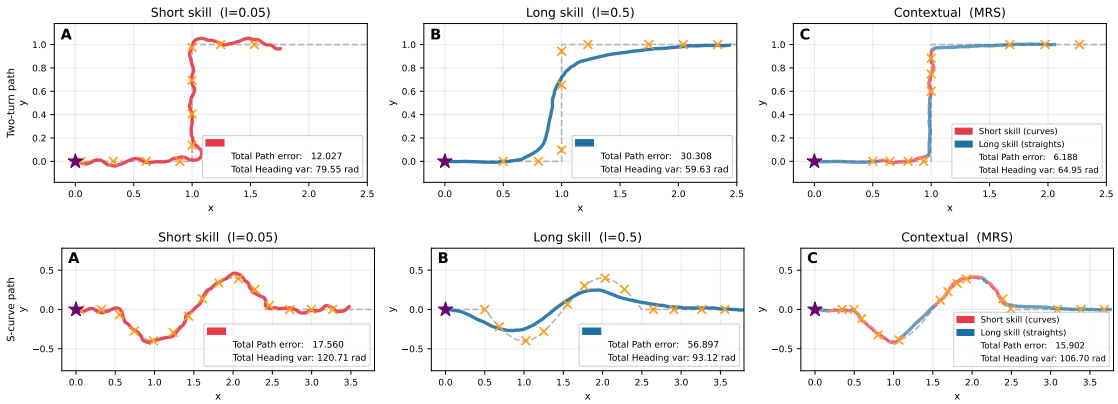


Figure 1: A point agent navigates two paths: a two-turn path (top) and an S-curve (bottom), by tracking subgoals sampled from future positions on the path with fixed additive noise. The short-skill agent (red) uses nearby subgoals: it follows curves precisely but accumulates jitter as the same absolute prediction noise produces large angular deviations when the target is close. The long-skill agent (blue) uses distant subgoals: it moves smoothly on straights but cuts corners and accumulates geometric error at transitions, as a fixed subgoal cannot capture the rapid heading change required at turns. The contextual agent (MRS, red/blue) selects short skills near turns and long skills on straights: it achieves lower total path error than either fixed alternative on the two-turn path, and correctly defaults to short skills throughout the continuously curved S-path. Crosses mark the assigned subgoal positions. Total path error and heading variance are reported in the legend. This illustrates that the optimal subgoal distance is both task- and state-dependent, and that no fixed skill length is universally optimal, motivating the adaptive multi-resolution selection in MRS. The skill switching rule here is hand-coded based on local curvature. In MRS, the meta-controller learns this selection from data.

path geometries. An agent that selects short skills near turns and long skills on straights outperforms either fixed alternative on heterogeneous paths, establishing that the optimal subgoal distance is both task- and state-dependent. This is similar to humans that utilize coarse-scale compound movements like full-leg motion, along with fine-grained foot placements for efficient running.

Motivated by these observations, we propose *Multi-Resolution Subgoals* (MRS), an HRL framework that addresses both problems. Rather than learning a single unconstrained goal representation, MRS learns multiple goal-prediction modules, each specialized to predict reachable states at a fixed temporal horizon. Because each module is conditioned on the current state and constrained to its temporal scale, its goal space is compact and concentrated on locally reachable states, and the manager can select precise local subgoals through short-horizon modules and smooth long-range targets through long-horizon modules. All modules share a common backbone and differ only in their resolution-specific heads, so performance gains are structural rather than capacity-driven. A learned meta-controller selects among resolution-specific skill policies based on the current state, enabling dynamic interleaving of fine- and coarse-grained behavior. The goal-prediction modules, skill policies, and meta-controller are jointly trained in a single end-to-end phase, with no task-specific tuning required.

MRS relates to two adjacent lines of prior work, discussed in detail in Sec. 5. Unsupervised skill discovery methods (Eysenbach et al., 2019; Sharma et al., 2020; Jiang et al., 2022) learn diverse behaviors without external rewards but impose no temporal structure on the skill space and cannot guarantee coverage of all local state transitions. Goal-conditioned HRL methods (Nachum et al., 2018b; Hafner et al., 2022; Vezhnevets et al., 2017; Zhang et al., 2020) use subgoal prediction but learn unconstrained goal representations. MRS retains the goal-conditioned framework but adds explicit temporal structure to the goal representation. HiPPO (Li et al., 2019) also addresses fixed temporal abstraction but randomizes skill length at each manager step for robustness, treating it as a fixed stochastic process rather than a learned, state-dependent decision. We implement MRS on top of the Director (Hafner et al., 2022) architecture, modifying only the goal representation and manager policy.

Our contributions are as follows:

1. We characterize a precision-smoothness tradeoff in subgoal-based HRL and show that unconstrained goal representations contribute to the known performance gap with monolithic agents on agility-demanding tasks. We further show that the optimal subgoal distance is both task- and state-dependent, motivating adaptive multi-resolution selection. (Fig. 1, Fig. 7, Fig. 6)

2. We propose MRS, which learns multiple goal-prediction modules each specialized to a fixed temporal horizon, with a shared backbone to minimize parameter overhead and a jointly trained meta-controller that selects among them based on current state, all within a single end-to-end training phase. (Sec. 3)

3. We empirically show that MRS consistently outperforms fixed-resolution baselines (Tab. 2), reduces the performance gap with non-HRL state-of-the-art on agility-demanding benchmarks (Tab. 1), and ablation studies confirm that the gains arise from temporal partitioning as an inductive bias. (Sec. 4)

2 Preliminaries

World Model. We build on the Recurrent State Space Model (RSSM) (Hafner et al., 2019), which encodes a state representation s_t using the observations o_t and previous state-actions (s_{t-1}, a_{t-1}) via a learned representation model $\texttt{repr}(s_t|o_t, s_{t-1}, a_{t-1})$. A dynamics model predicts future states $\texttt{dyn}(s_{t+1}|s_t, a_t)$ without requiring observations, enabling the agent to imagine trajectories entirely within the learned latent space. All policies in Director and MRS operate on s_t as their state representation.

Manager-Worker Decomposition. Director (Hafner et al., 2022) introduces a two-level hierarchy on top of the RSSM. The manager π_M proposes a subgoal s_g that updates every K steps. The worker π_W receives (s_t, s_g) at each step and outputs a primitive action a_t, trained with an intrinsic reward $R_t^W = \texttt{cosine_max}(s_t, s_g)$ that encourages it to move towards the subgoal. The manager is trained to maximize the sum of extrinsic task rewards and an exploratory bonus over each K-step interval; we describe the MRS exploratory objective in Sec. 3.4.

Goal Autoencoder. Directly predicting subgoals in the full RSSM state space poses a high-dimensional continuous-control problem for the manager. The Director therefore learns a goal autoencoder with encoder $\text{Enc}_\phi(z \mid s_t)$ and decoder $\text{Dec}_\phi(z) \approx s_t$, which compresses world model states into a compact categorical latent z using a mixture of categoricals as the prior. The manager operates in this latent space, sampling $z \sim \pi_M(z \mid s_t)$ and decoding to obtain $s_g = \text{Dec}_\phi(z)$. The autoencoder is trained on all states encountered in the replay buffer, without constraints on reachability or temporal distance from the current state, the limitation that MRS addresses.

Policy Optimization. Both, manager and worker, are trained as actor-critics optimized via policy gradients on trajectories imagined using the RSSM, using λ-returns for value estimation and entropy regularization to encourage exploration. MRS uses the same optimization procedure with a modified manager objective (Sec. 3.4).

3 Methodology

3.1 Skills as Constrained State Transitions

The Director's goal autoencoder learns to compress and reconstruct states unconditionally, without reference to the current state s_t. As a result, the manager can decode any point in the latent space into a subgoal, regardless of whether it is reachable from s_t within the worker's capabilities. We replace the goal autoencoder with a Conditional VAE (CVAE) that learns to model state transitions of a fixed length l. Given the current state s_t, the CVAE learns to encode and decode future states s_{t+l} that are l timesteps away under the current policy. Training pairs (s_t, s_{t+l}) are extracted from the replay buffer by taking states l steps apart within each trajectory. The CVAE encoder $\text{Enc}_\phi(z \mid s_t, s_{t+l})$ maps the transition to a latent skill variable z, and the decoder $\text{Dec}_\phi(s_t, z)$ reconstructs the target state conditioned on both s_t and z (Fig. 2a). Conditioning the

(a) Skill CVAE architecture

(b) Acting using Skill CVAE

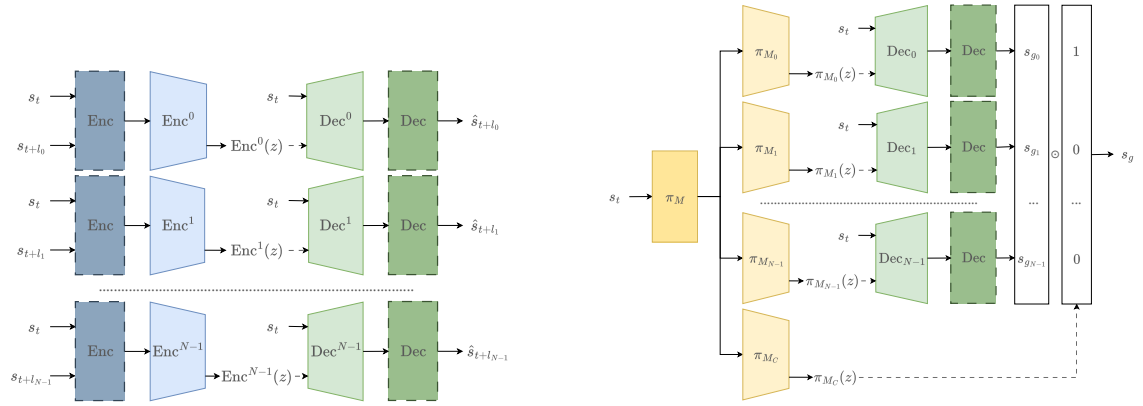
Figure 2: Illustrations of the abstract state transition-based control for the manager. *Dashed* arrows indicate sample propagation from the predicted distribution. **(a)** Skill CVAE, where the Encoder encodes initial and final states (s_t, s_{t+l}) to a latent skill space and the Decoder reconstructs the final state using the initial state s_t and a sampled skill variable z. **(b)** The manager predicts the latent skills and then uses the Decoder to generate goals for the worker.

decoder on s_t ensures that z encodes only the transition rather than the full state, keeping the latent space compact and structured relative to the current state. Therefore, the latent space corresponds to abstract state transitions possible from the current state s_t. And we refer to the latent samples z as skills. The CVAE is trained on the ELBO objective:

$$\mathcal{L}(\phi) = \|s_{t+l} - \text{Dec}_\phi(s_t, z)\|^2 + \beta \text{KL}\left[\text{Enc}_\phi(z \mid s_t, s_{t+l}) || p(z)\right], \quad z \sim \text{Enc}_\phi(z \mid s_t, s_{t+l}) \tag{1}$$

where $p(z)$ is a mixture of 8×8-dimensional categoricals used as the prior. At inference, the manager samples $z \sim \pi_M(z \mid s_t)$ and decodes to obtain the subgoal $s_g = \text{Dec}_\phi(s_t, z)$, constraining subgoal selection to the neighbourhood of states reachable from s_t in l steps (Fig. 2b). A single skill length l, however, cannot be universally optimal (Sec. 1); the appropriate subgoal distance varies by task and by state within a task. This motivates learning skills at multiple temporal resolutions simultaneously.

3.2 Multi-Resolution Skill Modules



(a) Learning Multi-Resolution CVAEs

(b) Acting using Multi-Resolution CVAEs

Figure 3: Architectures for learning and acting using Multi-Resolution skills ($l_i \in \{l_0, l_1, ..., l_{N-1}\}$). Dashed boundaries indicate shared layers. **(a)** Separate CVAEs are learned for each temporal resolution l_i. The Enc and Dec modules represent the common layers of the Encoders and the Decoders, respectively. Each Enc$_i$ is the resolution-specific encoder output layer, and each Dec$_i$ is the resolution-specific decoder input layer. **(b)** The manager's policy has $N + 1$ output heads. N skill heads π_{M_i} that predict the resolution-specific skill latents and choice head π_{M_C} that predicts an N-dimensional one-hot distribution. Samples from the skill latents are used to predict sub-goals using the respective Decoders, then the choice sample from π_{M_C} selects one of the sub-goals as s_g by gating.

We learn N CVAEs in parallel, indexed by resolution i, each specialized to a fixed temporal horizon l_i. We choose $l_i \in \{K, 2K, 4K, 8K\}$, where K is the manager's subgoal refresh interval. This geometric spacing is principled: $l = K$ is the shortest meaningful skill, corresponding to one full manager interval, and doubling gives logarithmically uniform coverage of the temporal horizon.

To prevent parameter growth from scaling with N, all CVAEs share a common backbone. All layers are shared except the encoder's last layer Enc^i_ϕ, and the decoder's first layer Dec^i_ϕ, which are resolution-specific (Fig. 3a). This means the parameter overhead of adding N resolutions is limited to N pairs of lightweight heads rather than N full models. Training extracts pairs (s_t, s_{t+l_i}) at each resolution from the replay buffer and optimizes the summed ELBO:

$$\mathcal{L}(\phi) = \sum_{i=0}^{N-1} \left\| s_{t+l_i} - \text{Dec}^i_\phi(s_t, z_i) \right\|^2 + \beta \text{KL}\left[\text{Enc}^i_\phi(z_i \mid s_t, s_{t+l_i}) || p(z)\right], \quad z_i \sim \text{Enc}^i_\phi \tag{2}$$

Shared layers receive gradients from all N resolutions simultaneously; resolution-specific heads receive gradients only from their own resolution. All CVAEs are trained jointly with the rest of the agent in a single end-to-end phase.

What about longer skills? As skill length l increases, the target state s_{t+l} becomes progressively less predictable from s_t as the agent can take many different trajectories in l steps. In the limit $l \to \infty$, the target state is statistically independent of the starting state, and the conditional distribution collapses to the marginal over all states. The ∞-skills can be implemented using an unconditional VAE trained on states from the replay buffer without conditioning on any s_t, producing subgoals that are independent of the current state by construction. Adding ∞ keeps the resolutions generic and makes the limiting case explicit.

Beyond its conceptual role, the ∞-skill addresses a practical failure mode in joint skill learning. The manager policy and the finite CVAEs are co-dependent: the CVAEs learn from trajectories generated by the manager, while the manager learns to use the skills the CVAEs represent, a circular dependency well-known to cause degenerate solutions in joint skill learning (Bacon et al., 2017; Eysenbach et al., 2019). If the finite CVAEs saturate early, converging to a small set of skills before the manager's policy has sufficiently explored them, the manager collapses into repeating those transitions, which in turn deprives the CVAEs of diverse training data. The ∞-skill breaks this cycle because it does not depend on any finite-length transition, allowing the manager to generate diverse subgoals and recover even when the finite-length CVAEs have collapsed. In practice, we found the agent initially relies on the ∞-skill and gradually shifts toward the finite skills as the CVAEs mature, a trend visible in the evolution of the choice distribution during training (Fig. 7). Results in Sec. 4 show that MRS remains competitive without the ∞-skill on most tasks, confirming that it functions as a robustness mechanism against early CVAE saturation rather than a core architectural requirement. The full resolution set used in all experiments is therefore $l_i \in \{K, 2K, 4K, 8K, \infty\}$, with $N = 5$ total skill modules including the ∞-skill.

3.3 The Meta-Controller

The manager policy has N skill heads and one choice head (Fig. 3b). Each skill head $\pi_{M_i}(z_i \mid s_t)$ predicts a latent skill distribution for its resolution, and the choice head $\pi_{M_C}(c \mid s_t)$ predicts a one-hot N-dimensional distribution over resolutions. At each manager step, all N skill heads sample latents z_i to decode to candidate subgoals (Eq. 3) and select from those via gating (Eq. 4).

$$s^i_g = \text{Dec}^i_\phi(s_t, z_i), \quad z_i \sim \pi_{M_i}(z_i \mid s_t) \tag{3}$$

$$s_g = \sum_{i=0}^{N-1} c_i \cdot s^i_g, \quad c \sim \pi_{M_C}(c \mid s_t) \tag{4}$$

The selected subgoal s_g is passed to the worker and held fixed for K steps. The consequences of this discrete one-hot gating for policy optimization are discussed in Sec. 3.4.

3.4 Training Objective

CVAE training. The N skill CVAEs are trained jointly on Eq. 2, updated from the replay buffer throughout training.

Manager and worker. Both policies are trained as actor-critics optimized via policy gradients on trajectories imagined through the RSSM, using λ-returns for value estimation and entropy regularization. The manager policy gradient takes the form:

$$\nabla \mathcal{J}(\pi_M) = \mathbb{E}\left[\Big(\sum_{i=0}^{N-1} c_i \underbrace{\nabla \log \pi_{M_i}(z_i \mid s_t)}_{\text{Skill head } i} + \underbrace{\nabla \log \pi_{M_C}(c \mid s_t)}_{\text{Choice head}} \Big) \cdot A_t \right] \tag{5}$$

where A_t is the advantage estimate and $c_i \in \{0, 1\}$ is the one-hot choice indicator. Because $c_i = 0$ for all unchosen heads at each step, the gradient with respect to skill head i's parameters is explicitly zeroed when that head is not selected, a direct consequence of the objective formulation rather than of the gating architecture. Each skill head, therefore, receives a gradient only from manager steps where it was chosen, keeping heads specialized to their own resolution and preventing gradient contamination across resolutions. Full derivations and loss definitions are provided in Appendix A.

Exploratory reward. To encourage the manager to discover the full diversity of state transitions in the environment across all temporal scales. Starting at state s_0, we reward the agent for executing transitions (s_0, s_t) that are not yet well-modeled by any skill module. This is implemented as the minimum CVAE reconstruction error across all N finite skills, selecting the error from whichever module currently best models the transition, and rewarding the manager for transitions that remain poorly explained by the best-fitting module:

$$R_t^{\text{Expl}} = \min_i \left\| s_t - \text{Dec}_\phi^i(s_0, z_i) \right\|^2, \quad z_i \sim \text{Enc}_\phi^i(z \mid s_0, s_t) \tag{6}$$

All components: skill CVAEs, skill policies, choice head, and worker, are optimized jointly throughout a single training phase with no task-specific tuning required.

3.5 Full Algorithms

Algorithms 1 and 2 illustrate the training and inference steps, respectively. See section C for further details, including hyperparameters, listed in Table 4.

Algorithm 1: Multi-Resolution Skill Training

Input: Collected trajectories $\mathcal{D} = \{\tau_1, ..., \tau_B\}$

```
// World Model Training
```
wm.train($\mathcal{D}$) // See Hafner et al. (2019)

```
// Multi-Resolution Skill Learning
```
$\mathcal{L}_{\text{skills}} \leftarrow [\,]$
for $l_i \in \mathcal{L}$ **do**
  $\{(s_t, s_{t+l_i})\} \leftarrow \text{ExtractStatePairs}(\mathcal{D}, l_i)$
  $\mathcal{L}_i \leftarrow \text{skill_loss}(s_t, s_{t+l_i})$ // CVAE loss (Eq. 1)
  $\mathcal{L}_{\text{skills}}.\text{append}(\mathcal{L}_i)$
update_skills(sum($\mathcal{L}_{\text{skills}}$))

```
// Policy Optimization via Imagination
```
$\mathcal{S}_{init} \leftarrow \{s_0 \mid s_0 \in \tau, \tau \in \mathcal{D}\}$ // Initial states
$\hat{\tau} \leftarrow$ `wm.imagine`$(\pi_{\text{MRS}}, \mathcal{S}_{init}, T)$ // Rollout imagined trajectories

```
// Reward Computation
```
$\hat{\tau}.r^{extr} \leftarrow r_{env}(\hat{\tau})$ // Environment reward
$\hat{\tau}.r^{expl} \leftarrow \text{expl_rew}(\hat{\tau})$ // Exploration reward (Eq. 6)
$\hat{\tau}.r^{goal} \leftarrow$ `cosine_max`$(\hat{\tau}.s_t, \hat{\tau}.s_g^{\lfloor t/K \rfloor})$ // Goal achievement reward

```
// Hierarchical Policy Update
```
$\mathcal{T}_W \leftarrow$ `split`$(\hat{\tau})$ // Worker-level transitions
$\mathcal{T}_M \leftarrow$ `abstract`$(\hat{\tau})$ // Manager-level abstractions
$\mathcal{L}(\pi_M), \mathcal{L}(v_M) = \text{manager_loss}(\mathcal{T}_M)$ // Eqs. 11,12
update_manager($\mathcal{L}(\pi_M), \mathcal{L}(v_M)$)
$\mathcal{L}(\pi_W), \mathcal{L}(v_W) = \text{worker_loss}(\mathcal{T}_W)$ // Eqs. 16,17
update_worker($\mathcal{L}(\pi_W), \mathcal{L}(v_W)$)

Algorithm 2: Multi-Resolution Skill Policy (π_{MRS})

Input: Observation o_t, Agent previous state $(t, s_{t-1}, a_{t-1}, s_g)$
Output: Action a_t, New agent state $(t+1, s_t, a_t, s_g)$

$s_t \leftarrow$ `wm`(o_t, s_{t-1}, a_{t-1}) // World model state computation

if $t \bmod K = 0$ **then**
  // Manager updates goal every K steps
  $(z_0, z_1, ..., z_{N-1}, c) \sim \pi_M(s_t)$ // Sample skill latent z and choice c
  $\{s_g^i\}_{i=0}^{N-1} \leftarrow \{\text{Dec}_\phi^i(s_t, z_i)\}_{i=0}^{N-1}$ // Generate candidate goals
  $s_g \leftarrow \sum_{i=0}^{N-1} c_i \cdot s_g^i$ // Select goal using choice vector c
else
  $s_g \leftarrow s_g$ // Persist previous goal

// Worker policy execution
$a_t \leftarrow \text{Worker}_\pi(s_t, s_g)$ // Generate action for current goal

Return $a_t, (t+1, s_t, a_t, s_g)$

4 Experiments

4.1 Experimental Setup

We evaluate MRS across three benchmark families that test distinct capabilities. The DeepMind Control Suite (DMC) (Tassa et al., 2018) provides six continuous control tasks (`walker_run`, `quadruped_run`, `cheetah_run`, `hopper_hop`, `cartpole_swingup`, `pendulum_swingup`) with dense rewards, testing whether MRS reduces

the agility gap relative to HRL baselines. Gymnasium-Robotics (de Lazcano et al., 2024) provides two sparse-reward (0 at goal, -1 otherwise) manipulation tasks (`fetch_push`, `fetch_pick_n_place`) that require precise goal-directed control over short horizons. AntMaze tests long-horizon (3000-step) sparse-reward (1 at goal, 0 otherwise) navigation, in which the agent must plan across extended sequences of subgoals without intermediate feedback.

We compare against three baselines. Director (Hafner et al., 2022) is the state-of-the-art HRL agent on which MRS is built, and represents the unconstrained subgoal HRL baseline. DreamerV3 (Hafner et al., 2023) is the state-of-the-art non-HRL agent, and represents the agility ceiling that HRL has historically failed to match. HiPPO (Li et al., 2019) is included as a temporal-abstraction baseline that executes subpolicies for random time-lengths, operates on privileged state observations, and uses on-policy RL.

Both MRS and Director are trained every 8th environment step using self-collected data from the replay buffer. DreamerV3 trains every 2nd step, making it approximately $4\times$ more compute-intensive. We use skill lengths $L = [64, 32, 16, 8, \infty]$ for all MRS experiments. All agents are evaluated across 4 seeds; we report the mean and standard deviation. Full hyperparameters and architecture details are provided in Appendix C.

4.2 Main Results

Table 1 reports final performance across all benchmarks. Fig. 4 shows learning curves for the representative DMC and Gym-Robotics tasks; full results for all tasks are provided in Appendix B.

Task	MRS (ours)	Director	DreamerV3	HiPPO
DeepMind Control Suite (locomotion)				
`walker_run`	$\mathbf{745 \pm 21}$	701 ± 19	$\underline{856 \pm 12.24}$	472 ± 85
`quadruped_run`	$\mathbf{925 \pm 19}$	719 ± 20	$\underline{937 \pm 15}$	572 ± 25
`cheetah_run`	$\mathbf{866 \pm 29}$	435 ± 129	$\underline{922 \pm 6}$	458 ± 30
`hopper_hop`	$\underline{\mathbf{511 \pm 56}}$	380 ± 55	487 ± 85	102 ± 37
`cartpole_swingup`	842 ± 6	684 ± 60	852 ± 3	$\underline{\mathbf{852 \pm 10}}$
`pendulum_swingup`	526 ± 297	474 ± 360	748 ± 58	$\underline{\mathbf{817 \pm 11}}$
Gymnasium-Robotics (manipulation, sparse reward)				
`fetch_push`	$\underline{\mathbf{-36 \pm 8}}$	-52 ± 9	-87 ± 7	-100 ± 0
`fetch_pick_place`	$\underline{\mathbf{-29 \pm 5}}$	-41 ± 8	-94 ± 3	-100 ± 0
AntMaze (long-horizon, sparse reward)				
Medium	$\underline{\mathbf{2299 \pm 296}}$	2035 ± 470	0 ± 0	0 ± 0
Large	$\underline{\mathbf{2224 \pm 395}}$	1982 ± 280	0 ± 0	0 ± 0

Table 1: Final performance as cumulative episodic rewards (mean $\pm$ std) on DMC, Gym-Robotics, and AntMaze. HiPPO uses privileged ground-truth state observations, whereas all other methods operate on pixels. Best HRL result **bold**; overall best <u>underlined</u>.

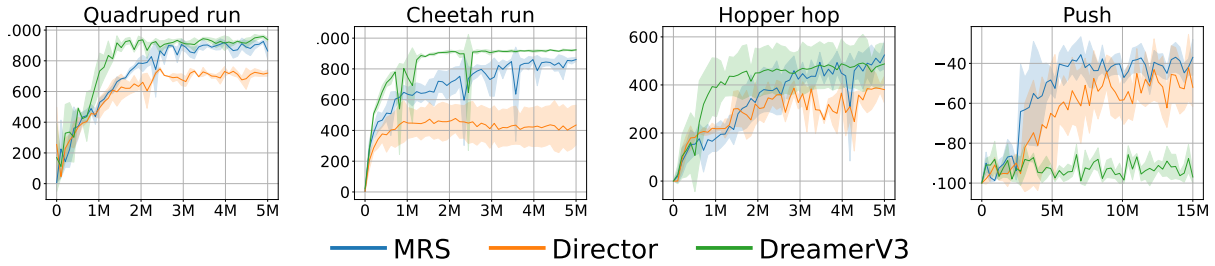


Figure 4: Episodic scores from MRS (ours), Director, and DreamerV3. The plot shows the total rewards (mean and standard deviation) received in an episode vs the environmental step at the representative DMC (first 3) and Gym-Robotics (Push) tasks. It can be seen that MRS noticeably improves the base model's performance in all cases while maintaining computational efficiency.

The results reveal a clear pattern across prior methods: Director excels on sparse-reward tasks requiring sustained planning, Gym-Robotics and AntMaze, but consistently underperforms on agility-demanding DMC tasks. DreamerV3 is competitive or superior on DMC but fails completely on Gym-Robotics, where its flat policy cannot reliably perform in sparse-reward tasks. Prior work, therefore, forces a choice between long-horizon competence and agility. MRS achieves competitive performance across all three regimes, matching or exceeding Director on Gym-Robotics and AntMaze, and reducing the gap with DreamerV3 on DMC. HiPPO performs well on the simple swingup tasks, but underperforms on the complex tasks. The `cheetah_run` gap (866 vs 435) warrants particular attention. The choice histogram (Fig. 7) does not show a clear preferred resolution for all states, but the t-SNE analysis (Fig. 6) reveals spatially structured selection where distinct state regions consistently prefer particular horizons. We interpret this as evidence that contextually appropriate subgoals in precision-critical states drive the gain, even when aggregate skill usage appears uniform; Director, restricted to a single goal distribution, cannot vary subgoal precision by local context. On AntMaze, MRS yields gains in both final performance and convergence speed. We note this as an observed benefit consistent with the higher sample efficiency of constrained subgoal prediction.

4.3 Ablations

Single-resolution baselines. We train four agents each using only a single fixed skill length from scratch, $l \in \{64, 32, 16\}$ and random selection, on `cheetah_run`, `quadruped_run`, and `hopper_hop`. Director serves as the ∞-only baseline. Table 2 reports the final performance. No single fixed resolution consistently matches MRS across tasks, and random selection performs slightly better than fixed skills. But MRS, with access to the full resolution set, consistently outperforms all fixed alternatives, confirming that the gains arise from multi-resolution coverage as a structural inductive bias rather than from the presence of any single skill length.

Task	**MRS**	Random	$l = 64$	$l = 32$	$l = 16$	$l = \infty$ **(Director)**
`cheetah_run`	866 ± 29	507 ± 55	511 ± 123	329 ± 41	484 ± 58	435 ± 129
`quadruped_run`	925 ± 19	856 ± 52	729 ± 70	791 ± 33	701 ± 63	719 ± 20
`hopper_hop`	511 ± 56	395 ± 37	195 ± 37	338 ± 87	378 ± 83	380 ± 55

Table 2: Default meta-controller setting compared against random selection and single-resolution baselines. Final performance (mean $\pm$ std, 3 seeds) for agents trained with a single fixed skill length versus full MRS. Director = ∞-only baseline.

Meta-controller contribution. Training with different skill settings can lead to differences in the training data for each setting. To verify if the its the training data that leads to performance differences or access to multi-resolution skills, we apply three alternative choice policies to a fully trained MRS agent (where all skill policies use the same training data) at evaluation time: (a) the default learned meta-controller, (b) uniform random skill selection at each manager step, and (c) exclusive use of each individual skill head $\{64, 32, 16, 8, \infty\}$. This setup compares the N CVAEs and N expert policy heads trained using the same data. Table 3 reports mean scores across 100 evaluation runs on all eight DMC tasks.

Choice	`walker_walk`	`walker_run`	`quad_walk`	`quad_run`	`cheetah_run`	`hopper_hop`	`cartpole_sw`	`pendulum_sw`
Default	$\mathbf{967 \pm 20}$	$\mathbf{771 \pm 12}$	$\mathbf{942 \pm 29}$	$\mathbf{900 \pm 38}$	$\mathbf{861 \pm 15}$	$\mathbf{497 \pm 61}$	$\mathbf{825 \pm 113}$	$\mathbf{526 \pm 424}$
Random	961 ± 20	744 ± 13	881 ± 39	855 ± 61	729 ± 86	394 ± 86	517 ± 269	306 ± 356
$l = 64$	965 ± 21	772 ± 12	929 ± 28	887 ± 30	784 ± 40	461 ± 53	678 ± 253	436 ± 405
$l = 32$	962 ± 20	739 ± 11	937 ± 30	895 ± 40	687 ± 86	433 ± 91	614 ± 302	322 ± 372
$l = 16$	958 ± 23	734 ± 15	787 ± 52	749 ± 208	483 ± 35	221 ± 240	474 ± 272	109 ± 312
$l = 8$	953 ± 30	714 ± 23	698 ± 64	298 ± 403	544 ± 100	165 ± 187	231 ± 89	40 ± 196
$l = \infty$	952 ± 25	728 ± 67	879 ± 37	842 ± 35	846 ± 17	401 ± 48	573 ± 248	501 ± 430

Table 3: Meta-controller ablation. Mean episodic reward across 100 evaluation runs. All conditions use the same trained MRS skill representations; only the choice mechanism varies. Best result per task in **bold**.

The default meta-controller achieves higher average performance than random selection and the best individual skill across tasks. The margin over random selection confirms that the meta-controller has learned a structured, non-uniform selection policy (Fig. 6). Together, these results confirm that learned selection adds value beyond that provided by the individual skills.

Exploration objective. We compare four agents after an exploration phase followed by 1M steps of task fine-tuning on four DMC tasks: MRS, MRS with a vanilla exploratory loss (replacing the min-CVAE reconstruction error with Director's unconditional VAE reconstruction error), ReST (Jiang et al., 2022), and DIAYN (Eysenbach et al., 2019). MRS operates on visual observations, whereas ReST and DIAYN use privileged ground-truth states. MRS achieves competitive or superior fine-tuning performance and substantially faster convergence (Fig. 5). MRS, with our exploratory objective, outperforms the vanilla training objective, indicating that the MRS exploratory objective alone can help learn useful state transitions.

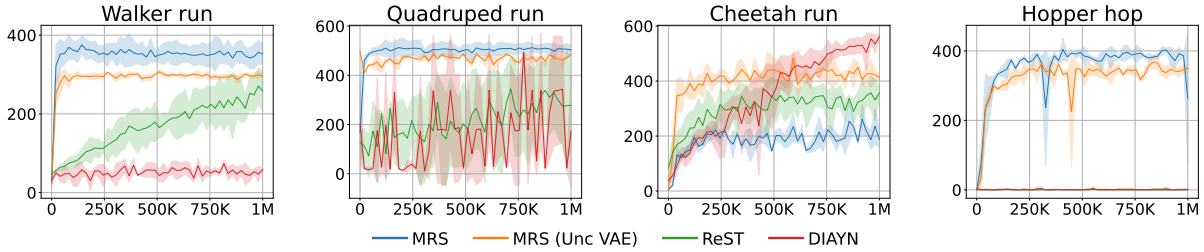


Figure 5: Performance comparison of agents fine-tuned for tasks after an exploration phase (3 seeds per experiment). The graphs show total episodic reward (mean and standard deviation) against the global steps. The plots compare: MRS, MRS using exploratory rewards from the unconditional VAE, ReST, and DIAYN. Our agent is trained every 8-th step using image inputs, while DIAYN/ReST trains every step using the internal environmental proprioceptive state.

4.4 Qualitative Analysis

Skill selection structure. Fig. 6 shows t-SNE projections of world model states visited during evaluation, colored by the skill length chosen by the meta-controller. Across tasks, skill selection is spatially structured: distinct regions of the latent state space show consistent preferences for particular temporal horizons rather than uniform or random assignment across the embedding. We quantify this with cluster purity: the fraction of points within each k-means cluster sharing the same dominant skill label, reporting a mean purity of **0.6** across tasks, substantially above the random baseline of $1/N = 0.20$. This confirms that the meta-controller has learned a state-dependent selection policy with genuine structure, and supports the interpretation in Sec. 4.2 that contextually appropriate subgoal selection in specific state regions drives the significant performance gains observed on tasks such as `cheetah_run`.

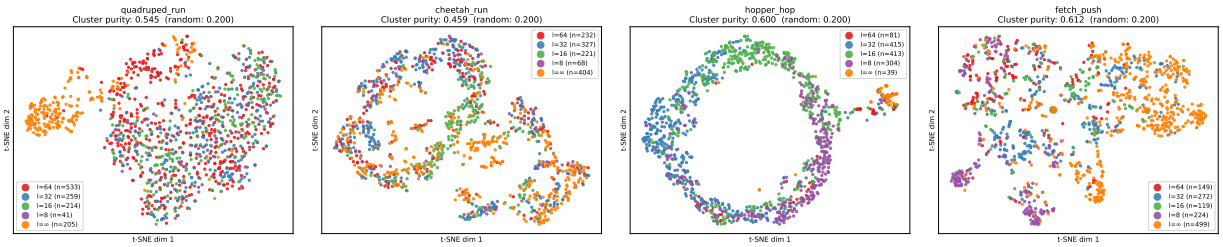


Figure 6: T-SNE projections of the environmental states, colored by the CVAE index i. The topological state clusters in the 2D space coincide with the color clusters, indicating that the agent prefers specific skill resolutions in specific state types. To measure cluster purity, we apply k-means clustering to the t-SNE projections, and purity is the ratio of the number of points assigned to the dominant skill to the total number of points in each cluster. Thus, following a pattern rather than being random.

Training dynamics of skill selection. Fig. 7 shows the evolution of the choice distribution during training across all DMC and Gym-Robotics tasks. A consistent pattern emerges: agents begin with a strong preference for the ∞-skill, then gradually shift toward finite skills as the CVAEs mature. The initial ∞-preference is consistent with the chicken-and-egg motivation in Sec. 3.2: early in training, finite CVAEs have not yet learned reliable transition representations, and the ∞-skill provides a stable source of exploratory subgoals. As training proceeds and the CVAEs converge, the meta-controller progressively exploits the precision advantages of finite skills.

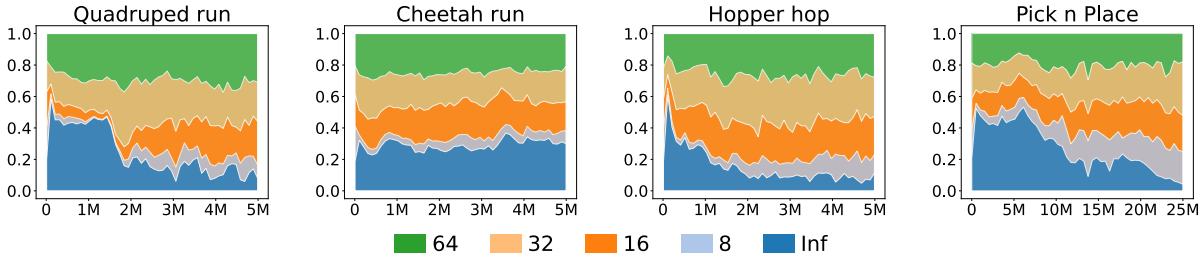


Figure 7: Stream graphs showing the evolution of the choice distribution during training averaged across 4 seeds. A trend can be observed in some tasks where the manager initially focuses on ∞-length skills but gradually shifts to temporally constrained skills.

Samples from trained models. We provide an extensive list of samples generated using a trained agent:

- Interesting behaviors like flips, maintaining headstand, and tumbling using the exploratory objective (Sec. D).
- Sample subgoal using each skill CVAE trained using task and exploratory rewards (Sec. E).
- Visual observations aggregate by choice (Sec. F).

5 Related Work

Subgoal-based hierarchical RL. Hierarchical reinforcement learning formalizes temporal abstraction by allowing agents to select temporally-extended actions rather than only primitive actions Sutton et al. (1999); Precup (2000); Barto & Mahadevan (2003); Botvinick et al. (2009); Pateria et al. (2021). Early feudal decompositions Dayan & Hinton (1992) introduced the manager-worker structure that underlies modern HRL; neural instantiations include FeUdal Networks Vezhnevets et al. (2017) and h-DQN Kulkarni et al. (2016), which demonstrated that learned subgoal hierarchies could tackle tasks with sparse rewards. The Option-Critic architecture Bacon et al. (2017) extended this line by learning option policies and termination conditions end-to-end without manually specified subtasks. Goal-conditioned methods such as HIRO Nachum et al. (2018b) and Director Hafner et al. (2022) train a higher-level policy to propose subgoal states and a lower-level policy to reach them, enabling long-horizon planning from pixels without access to privileged state information. A parallel line of work focuses on improving the quality of subgoal representations directly: HRAC Zhang et al. (2020) constrains subgoals to an adjacency-based reachable set, and HLPS Wang et al. (2024) introduces probabilistic subgoal representations to improve robustness under stochasticity. These methods address the unconstrained goal space in isolation but do not partition the goal representation by temporal horizon, and therefore cannot resolve the precision-smoothness tradeoff that MRS identifies as a structural source of the HRL agility gap.

Temporal abstraction and variable commitment. A distinct line of work addresses how long a higher-level commitment should be maintained. Fixed-commitment methods Florensa et al. (2017); Gupta et al. (2018) hold a skill active for a predetermined number of steps, sacrificing adaptability for simplicity. Timed subgoals Gürtler et al. (2021) associate each subgoal with a fixed commitment window, improving temporal structure but providing no mechanism to vary commitment based on local state context. HiPPO Li et al. (2019) explicitly tackles fixed temporal abstraction by randomizing the commitment length at each manager step, sampled from a categorical distribution over predetermined periods; this improves robustness

to environment changes by averaging over the nuisance of skill duration rather than learning to exploit it. STRAW Vezhnevets et al. (2016) learns macro-actions of variable length by predicting when to replan, but does so reactively rather than by maintaining a structured set of temporal resolutions. More recent work introduces reachability-aware abstractions Zadem et al. (2024) that refine the goal space iteratively based on k-step reachability relations, providing spatial structure but without an explicit temporal partitioning mechanism. MRS takes a different stance from all of these: rather than fixing, randomizing, or reactively terminating temporal commitments, MRS maintains a discrete set of skills at geometrically spaced temporal horizons and learns a meta-controller that selects among them based on current state, treating temporal resolution as a structured, learnable signal rather than a nuisance variable.

Unsupervised skill discovery. A large body of work discovers diverse skills without external rewards by maximizing mutual information (MI) between a latent skill variable and states or trajectories. Representative methods include DIAYN Eysenbach et al. (2019), which maximizes MI between skill codes and visited states; DADS Sharma et al. (2020), which maximizes MI between skills and state transition dynamics; VALOR Achiam et al. (2018) and OPAL Ajay et al. (2021), which extend MI objectives to full trajectory segments; and InfoGAN-based approaches Kurutach et al. (2018). Sequential coverage methods such as ReST Jiang et al. (2022) build on this by training skills sequentially to increase state-space coverage. The principal advantage of these methods is behavioral diversity without reward engineering; their principal limitation is that mutual-information objectives impose no temporal structure on the skill space and provide no mechanism for state-conditioned prediction of reachable future states at a specific horizon. Because they do not condition on s_t or constrain skill length, they cannot concentrate representational capacity on locally reachable transitions as MRS's CVAEs do. MRS lies between unsupervised skill discovery and goal-conditioned HRL: it retains the goal-conditioned framework to enable temporal structure, while its exploratory objective draws on the diversity-seeking motivation of skill-discovery methods. This connection is evaluated directly in Sec. 4.3, where MRS's exploration phase is compared against DIAYN and ReST after fine-tuning on downstream tasks.

6 Discussion

The results establish that the HRL agility gap is not primarily a capacity or training problem, because MRS uses comparable parameters to Director and is trained at the same frequency, yet consistently reduces the gap with DreamerV3 on agility-demanding tasks while retaining competitiveness on long-horizon and sparse-reward benchmarks. The ablations support a more specific conclusion: a representational problem arising from the absence of a structure in the goal space is one reason for the gap. Imposing a structure, like temporal, through fixed-horizon CVAEs and a learned discrete gate, without modifying the world model, worker, or training procedure, is sufficient to produce the observed gains. This has implications beyond MRS: it suggests that other subgoal-based HRL architectures may benefit from similar temporal partitioning of their goal or option spaces, independent of the specific world model or policy optimization algorithm used. There exist two limitations in the current instantiation of MRS. First, MRS inherits the Director's dependence on a reliable RSSM. Tasks where world model quality is low, the CVAEs will also struggle, and the precision advantage of constrained subgoal prediction may be reduced. Second, the resolution set $\{K, 2K, 4K, 8K, \infty\}$ is fixed across all tasks; while the geometric spacing is principled, the optimal set may be task-dependent.

Three directions follow naturally from these observations. First, adaptive resolution discovery learning the resolution set rather than fixing it, for instance by initializing with a dense set of horizons and pruning underused skills during training, would remove the remaining hand-specified structure from MRS and potentially improve performance on tasks where the geometric spacing is suboptimal. Second, the architecture naturally accommodates hybrid skills that combine learned CVAEs with analytic or hand-specified expert primitives, which may be valuable in robotics settings where certain motion primitives are known a priori (like moving the end-effector along a straight line) and do not need to be learned from data. Third, the multi-head policy gradient formulation is architecture-agnostic: the discrete choice gate and per-head gradient separation could be applied to other HRL frameworks beyond Director, including options-based hierarchies Bacon et al. (2017) and language-conditioned managers Jiang et al. (2019), extending the benefits of temporal partitioning to a broader class of methods.

References

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

Anurag Ajay, Aviral Kumar, Pulkit Agrawal, Sergey Levine, and Ofir Nachum. {OPAL}: Offline primitive discovery for accelerating offline reinforcement learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=V69LGwJ0lIN.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

Andrew G Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 13(1):41–77, 2003.

Matthew M Botvinick, Yael Niv, and Andew G Barto. Hierarchically organized behavior and its neural foundations: A reinforcement learning perspective. *Cognition*, 113(3):262–280, 2009.

Peter Dayan and Geoffrey E Hinton. Feudal reinforcement learning. *Advances in neural information processing systems*, 5, 1992.

Rodrigo de Lazcano, Kallinteris Andreas, Jun Jet Tai, Seungjae Ryan Lee, and Jordan Terry. Gymnasium robotics, 2024. URL http://github.com/Farama-Foundation/Gymnasium-Robotics.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SJx63jRqFm.

Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1704.03012*, 2017.

Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Meta-reinforcement learning of structured exploration strategies. *Advances in neural information processing systems*, 31, 2018.

Nico Gürtler, Dieter Büchler, and Georg Martius. Hierarchical reinforcement learning with timed subgoals. *Advances in Neural Information Processing Systems*, 34:21732–21743, 2021.

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

Danijar Hafner, Kuang-Huei Lee, Ian Fischer, and Pieter Abbeel. Deep hierarchical planning from pixels. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 26091–26104. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/a766f56d2da42cae20b5652970ec04ef-Paper-Conference.pdf.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Yiding Jiang, Shixiang Shane Gu, Kevin P Murphy, and Chelsea Finn. Language as an abstraction for hierarchical deep reinforcement learning. *Advances in neural information processing systems*, 32, 2019.

Zheyuan Jiang, Jingyue Gao, and Jianyu Chen. Unsupervised skill discovery via recurrent skill training. *Advances in Neural Information Processing Systems*, 35:39034–39046, 2022.

Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

Thanard Kurutach, Aviv Tamar, Ge Yang, Stuart J Russell, and Pieter Abbeel. Learning plannable representations with causal infogan. *Advances in Neural Information Processing Systems*, 31, 2018.

Alexander C Li, Carlos Florensa, Ignasi Clavera, and Pieter Abbeel. Sub-policy adaptation for hierarchical reinforcement learning. *arXiv preprint arXiv:1906.05862*, 2019.

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

Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31, 2018b.

Shubham Pateria, Budhitama Subagdja, Ah-hwee Tan, and Chai Quek. Hierarchical reinforcement learning: A comprehensive survey. *ACM Comput. Surv.*, 54(5), jun 2021. ISSN 0360-0300. doi: 10.1145/3453160. URL `https://doi.org/10.1145/3453160`.

Doina Precup. *Temporal abstraction in reinforcement learning*. University of Massachusetts Amherst, 2000.

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

Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction second edition. *Adaptive computation and machine learning: The MIT Press, Cambridge, MA and London*, 2018.

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

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Alexander Vezhnevets, Volodymyr Mnih, Simon Osindero, Alex Graves, Oriol Vinyals, John Agapiou, et al. Strategic attentive writer for learning macro-actions. *Advances in neural information processing systems*, 29, 2016.

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

Vivienne Huiling Wang, Tinghuai Wang, Wenyan Yang, Joni-Kristian Kämäräinen, and Joni Pajarinen. Probabilistic subgoal representations for hierarchical reinforcement learning. *arXiv preprint arXiv:2406.16707*, 2024.

Mehdi Zadem, Sergio Mover, and Sao Mai Nguyen. Reconciling spatial and temporal abstractions for goal representation. *arXiv preprint arXiv:2401.09870*, 2024.

Tianren Zhang, Shangqi Guo, Tian Tan, Xiaolin Hu, and Feng Chen. Generating adjacency-constrained subgoals in hierarchical reinforcement learning. *Advances in neural information processing systems*, 33: 21579–21590, 2020.

A Derivation of Policy Gradients for MRS

We first decompose the action prediction process to derive the policy gradient to train the manager and worker policies. Let an MRS agent be in state s_t at step t. Every K-th step, the manager refreshes the worker's goal. For clarity, let the abstract step be indexed by k, then at each abstract step ($t = kK$):

1. Sample skill latents from the skill heads: $z_{k,0}, z_{k,1}, ..., z_{k,N-1} \sim \Pi_{i=0}^{N-1} \pi_{M_i}(z_{k,i}|s_{kK})$.
2. Sample a choice variable: $c_k \sim \pi_{M_C}(c_k|s_{kK})$.
3. Compute the selected subgoal: $s_g^k = \sum_{i=0}^{N-1} c_{k,i} \cdot \mathrm{Dec}_\phi^i(s_{kK}, z_{k,i})$.
4. Predict the environmental actions using worker: $\pi_W(a_t|s_t, s_g^k)$

Thus, the trajectory probability that starts at s_0 can be written as:

$$p(\tau) = p(s_0) \prod_{k=0}^{\lfloor T/K \rfloor - 1} \underbrace{\pi_{M_C}(c_k|s_{kK}) \prod_{i=0}^{N-1} \pi_{M_i}(z_{k,i}|s_{kK})^{c_{k,i}}}_{\text{Manager}} \prod_{t=0}^{T-1} \underbrace{\pi_W(a_t|s_t, s_g^{\lfloor t/K \rfloor})}_{\text{Worker}} \cdot \underbrace{p_T(s_{t+1}|a_t, s_t)}_{\text{State transition}} \tag{7}$$

The components of the equation can be read as: the manager predicts the skills $(z_{k,0}, z_{k,1}, ..., z_{k,N-1})$ and choice c_k for every abstract step k, the worker predicts the action a_t at each step t using the subgoal $s_g^{\lfloor t/K \rfloor}$ for the duration, and the environmental state transition p_T. Here, the exponent $c_{k,i}$ collapses the probability $\pi_{M_i}(z_{k,i}|s_{kK})$ of the unselected skill at step kK head to 1, since it does not affect the trajectory.

We follow the policy gradient derivation from Sutton & Barto (2018). The aim is to compute $\nabla_\theta J$, where $J = \mathbb{E}_\tau[R(\tau)]$ is the expected reward and θ are the policy parameters. Using the standard log-derivative trick (Sutton & Barto (2018)), the objective can be written as maximizing the trajectory log-probability weighted by the expected reward:

$$\nabla_\theta J = \mathbb{E}_\tau[R(\tau) \cdot \nabla_\theta \log p(\tau)]$$

The gradient of the trajectory log-probability w.r.t. the manager parameters M is:

$$\nabla_M \log p(\tau) = \sum_{k=0}^{\lfloor T/K \rfloor - 1} [\nabla_M \log \pi_{M_C}(c_k|s_{kK}) + \sum_{i=0}^{N-1} c_{k,i} \nabla_M \log \pi_{M_i}(z_{k,i}|s_{kK})]$$

Therefore, the policy-gradient objective can be written as:

$$\nabla_M J = \mathbf{E}_\tau[R(\tau) \cdot \sum_{k=0}^{\lfloor T/K \rfloor - 1} [\nabla_M \log \pi_{M_C}(c_k|s_{kK}) + \sum_{i=0}^{N-1} c_{k,i} \nabla_M \log \pi_{M_i}(z_{k,i}|s_{kK})]]$$

Given these policy gradients, we construct the losses for each head as the sum of the policy gradient objective and an entropy maximization objective (Eq. 10,9), and sum them for the total loss (Eq. 11).

$$G_k^\lambda = R_k + \gamma((1-\lambda)v_M(s_{kK}) + \lambda G_{k+1}^\lambda) \tag{8}$$

$$\mathcal{L}(\pi_{M_c}) = -\mathbb{E}_\tau \sum_{k=0}^{\lfloor T/K \rfloor - 1} \log \pi_{M_c}(c_k|s_{kK}) \cdot A_k + \eta \mathbb{H}[\pi_{M_C}(c_k|s_{kK})] \tag{9}$$

$$\mathcal{L}(\pi_{M_i}) = -\mathbb{E}_\tau \sum_{k=0}^{\lfloor T/K \rfloor - 1} c_{k,i} \cdot \log \pi_{M_i}(z_{k,i}|s_{kK}) \cdot A_k + \eta \mathbb{H}[\pi_{M_i}(z_{k,i}|s_{kK})] \tag{10}$$

$$\mathcal{L}(\pi_M) = \mathcal{L}(\pi_{M_c}) + \sum_{i=0}^{N-1} \mathcal{L}(\pi_{M_i}) \tag{11}$$

$$\mathcal{L}(v_M) = \mathbb{E}_\tau \sum_{k=0}^{\lfloor T/K \rfloor - 1} (v_M(s_{kK}) - G_k^\lambda)^2 \tag{12}$$

Where $A_k = G_k^\lambda - v_M(s_{kK})$ is the lambda returns estimated using abstract trajectories (Eq. 8), v_M is the critic (Eq. 12). The policy maximizes the advantage $G_k^\lambda - v_M(s_{kK})$ instead of directly maximizing estimated returns. Weighted entropic losses $\mathbb{H}[\cdot]$ encourage adequate exploration prior to convergence. The manager estimates separate returns, learns separate critics, and computes separate advantages for the external rewards R_k^{Extr} and exploratory rewards R_k^{Expl} (Eq. 6). The rewards are summed over the K-length trajectory chunks. The total advantage is the weighted sum of the exploratory and external advantages (($[1.0, 0.1]$) in our case).

B Full Results

B.1 Training Curves

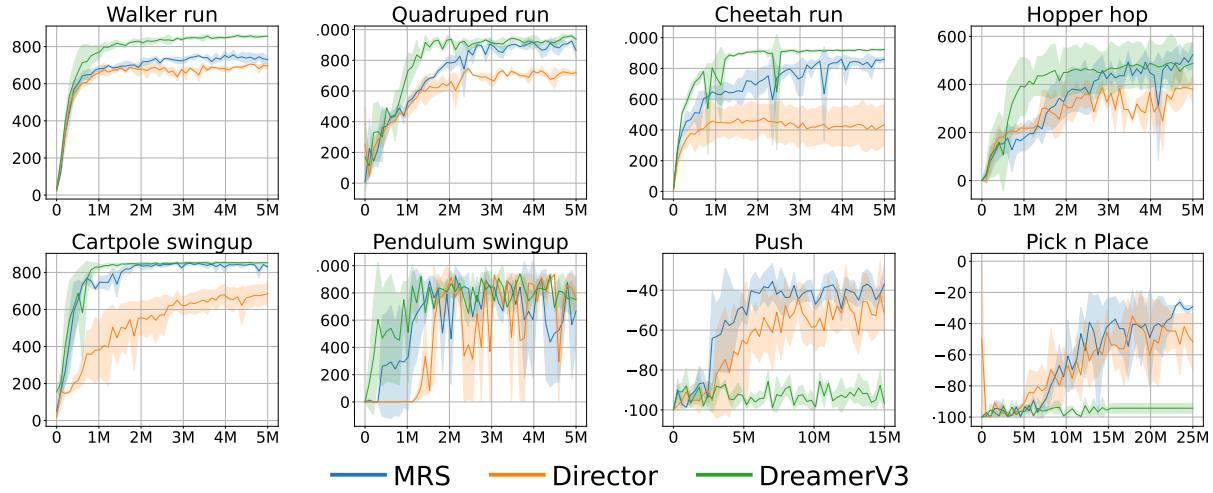


Figure 8: Episode scores from MRS (ours), Director, and DreamerV3. The plot shows the total rewards (mean and standard deviation) received in an episode against the environmental step. Both the Director and MRS use the same common hyperparameters. The first *six* tasks are from the DMC suite, and the last *two* (Push, and Pick n Place) are from the Gymnasium-Robotics suite. It can be seen that MRS noticeably improves the base model's performance in all cases while maintaining computational efficiency.

B.2 t-SNE Projections

Figure 9 shows the t-SNE projections of the states colored by the skill (temporal distance) choice preference. Since the clusters created by the state embeddings are marked by distinct colors, this indicates that the choice is not random but strategic.

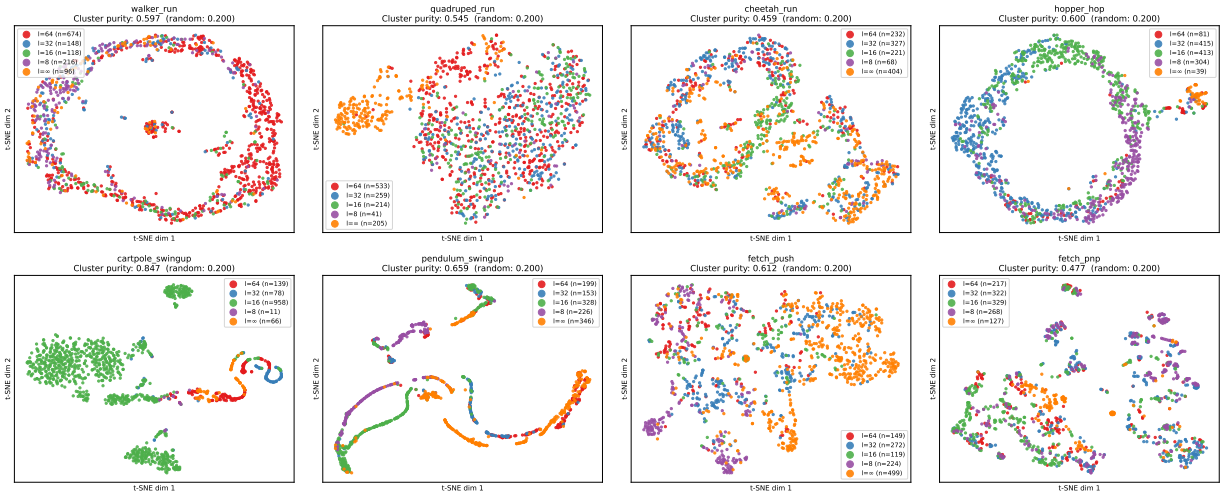


Figure 9: T-SNE projections of the states, colored by the chosen skill's index for the DMC and Gym-Robotics tasks. To measure cluster purity, we apply k-means clustering (with the number of clusters chosen to maximize the silhouette score), and the purity is the ratio of the number of points belonging to the dominant skill to the total points in each cluster. Thus, following a pattern instead of being random.

B.3 Choice Streamgraphs

Figure 10 shows the distribution of skill preference during training. A common trend across some tasks is that the agent starts with inf-length subgoals and then moves towards finite-length skills. While it continued to be distributed uniformly for some. It should be noted that uniform distribution does not imply random choice; the skill selection is strategic as shown in 9.

Figure 10: Stream graphs showing the evolution of the choice distribution during training. A trend can be observed in some tasks where the manager initially focuses on ∞-length skills but gradually shifts to temporally constrained skills.

C Architecture & Training Details

C.1 Worker

The worker is trained using K-step imagined rollouts ($\kappa \sim \pi_W$). Given the imagined trajectory κ, the rewards for the worker R_t^W are computed as the `cosine_max` similarity measure (Eq. 13) between the trajectory states s_t and the prescribed worker goal s_g (Eq. 14). First, discounted returns G_t^λ are computed as n-step lambda returns (Eq. 15). Then the Actor policy is trained using the SAC objective (Eq. 16) and the Critic is trained to predict the discounted returns (Eq. 17). The entropy for the worker and the manager is weighted to maintain a target entropy.

$$\texttt{cosine_max}(a, b) = ||a \cdot b|| / \max(||a||, ||b||)^2 \tag{13}$$

$$R_t^W = \texttt{cosine_max}(s_t, s_g) \tag{14}$$

$$G_t^\lambda = R_{t+1}^W + \gamma_L((1-\lambda)v(s_{t+1}) + \lambda G_{t+1}^\lambda) \tag{15}$$

$$\mathcal{L}(\pi_W) = -\mathbb{E}_{\kappa\sim\pi_W} \sum_{t=0}^{H-1} \left[(G_t^\lambda - v_W(s_t)) \ln \pi_W(z|s_t) + \eta \mathrm{H}[\pi_W(z|s_t)]\right] \tag{16}$$

$$\mathcal{L}(v_W) = \mathbb{E}_{\kappa\sim\pi_W} \left[\sum_{t=0}^{H-1} (v_W(s_t) - G_t^\lambda)^2\right] \tag{17}$$

C.2 Implementation Details

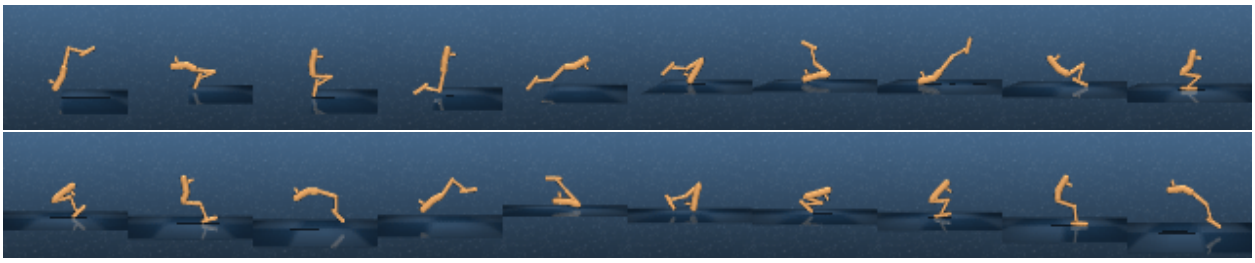
We implement two functions: `policy` (Alg. 2) and `train` (Alg. 1), using the hyperparameters shown in Table 4. The functions are implemented in Python/Tensorflow using XLA JIT compilation. The experiments on average take 2 days to run 5M steps on an NVIDIA RTX 5000.

Name	Symbol	Value
Train batch size	B	16
Replay data length	-	64
Worker abstraction length	K	8
Explorer Imagination Horizon	T	16
Return Lambda	λ	0.95
Return Discount	γ	0.99
Skill resolutions	L	$\{64, 32, 16, 8, \infty\}$
Target entropy	η	0.5
KL loss weight	β	1.0
RSSM deter size	-	1024
RSSM stoch size	-	32×32
Optimizer	-	Adam
Learning rate (all)	-	10^{-4}
Adam Epsilon	-	10^{-6}
Weight decay (all)	-	10^{-2}
Activations	-	LayerNorm + ELU
MLP sizes	-	4×512
Train every	-	8
Prallel Envs	-	4

Table 4: Agent Hyperparameters

D Behaviors Learned via Exploration

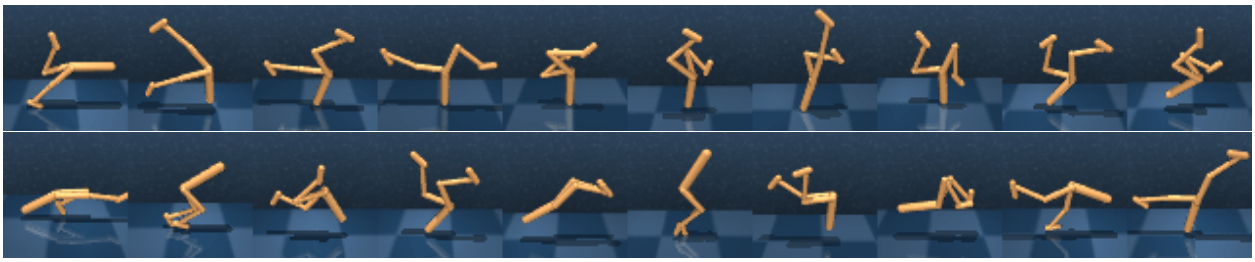
We observed several interesting behaviors exhibited by the MRS agent during training solely with the exploratory loss, including front flips, back flips, and jumps. The intrinsic exploratory loss encourages the agent to perform novel state transitions (Sec. 3.4). Fig. 11 shows some of the learned movements.



(a) Hopper learns to use a front flip to stand, and back flips.

(b) Cheetah learns to leap forward and perform perfect back flips.

(c) Quadruped learning side rolls and walking on two legs.



(d) Walker trying to maintain a headstand repeatedly and fast-forward tumbling using head and legs.

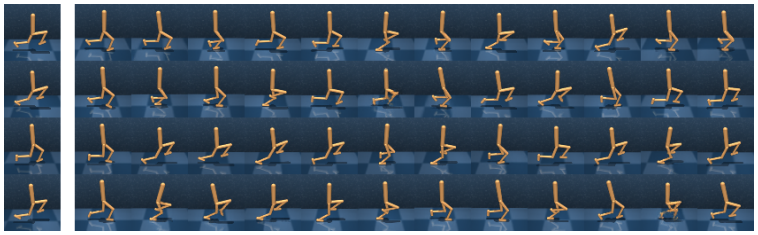
Figure 11: Samples of some movements learned and regularly performed by the agent optimized only for the exploratory loss.

E Sample Goals using Skill CVAEs

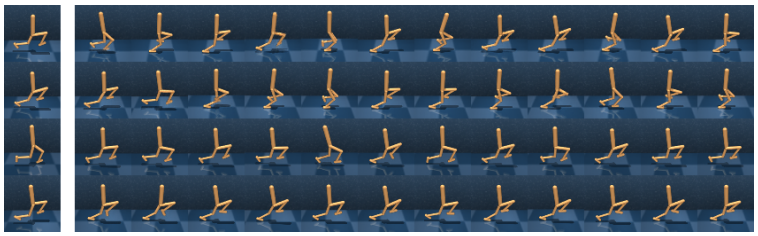
We generate some sample goals using each of the Skill CVAEs individually. The goals are generated using a uniform prior $p(z)$ for the skills, and initial states s_t sampled from the replay database. We use skills with temporal resolutions $[64, 32, 16, 8, \infty]$, but we omit the ∞ length skills, as they correspond to simply learning all states independently and are not our contribution.

E.1 Default Objective

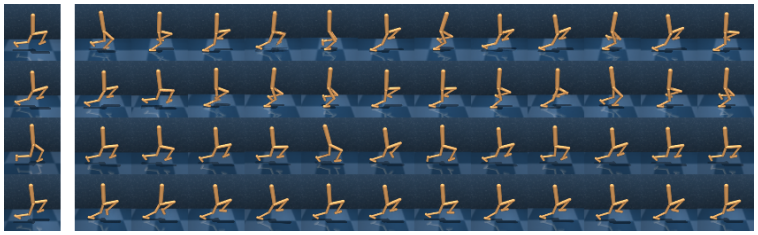
The agent is trained using the default objective (weighted external and exploratory advantages). Since we use a strong bias towards the external reward ($[1.0, 0.1]$), the learned skills are biased toward goal states that are more appropriate for the objective. We sample the goals for tasks: `walker_run` (Fig. 12), `quadruped_run` (Fig. 13), `cheetah_run` (Fig. 14), and `hopper_hop` (Fig. 15).



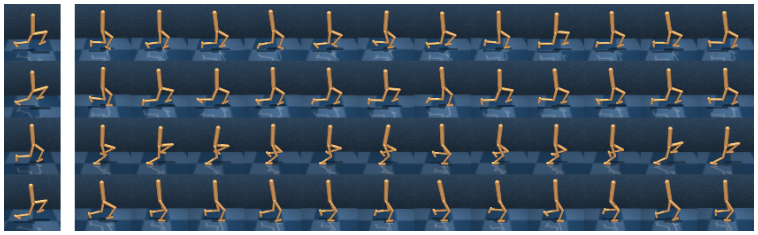
(a) 64 length.



(b) 32 length.

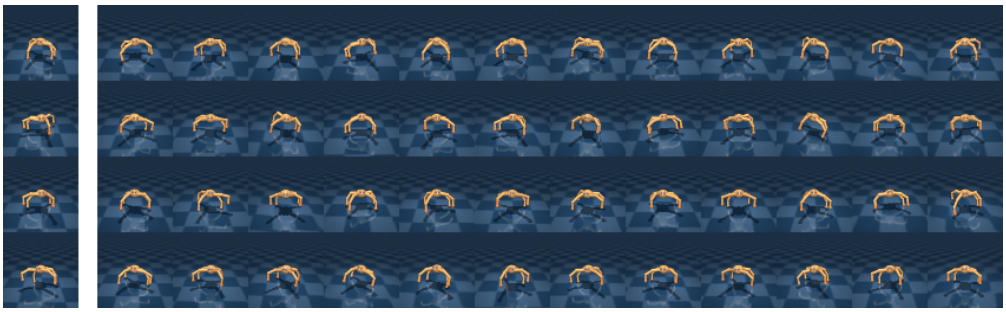


(c) 16 length.

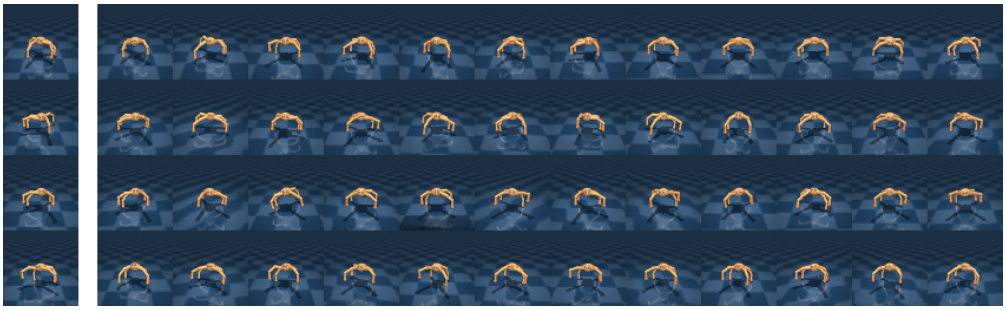


(d) 8 length.

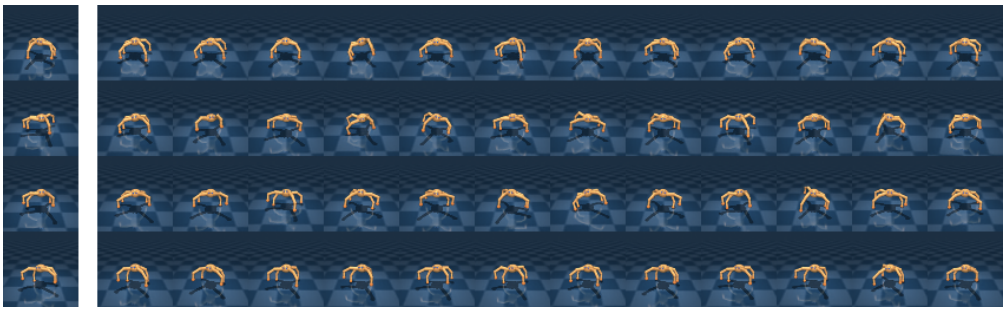
Figure 12: Sample goals from the `walker_run` task learned using the external and the exploratory rewards. The images on the left show the current state s_t, and the remaining images show the goal options generated by different skill CVAEs.



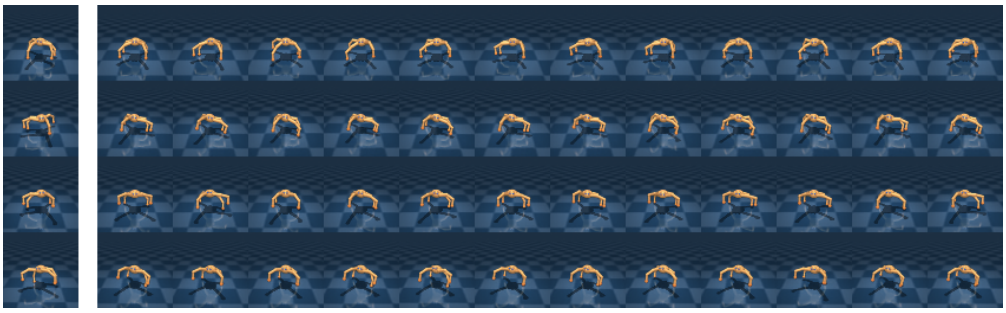
(a) 64 length.



(b) 32 length.



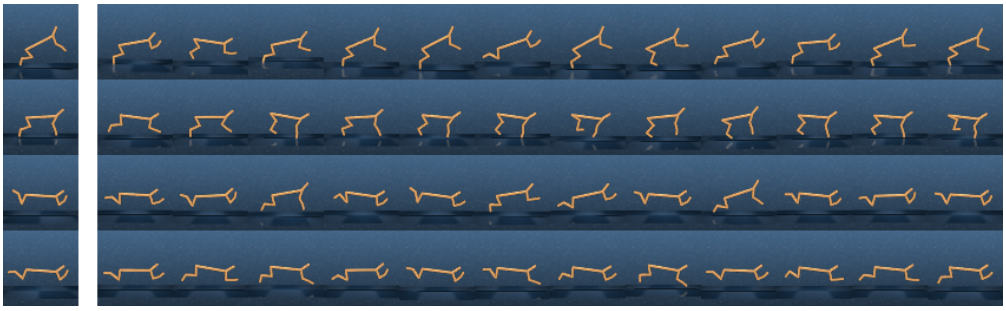
(c) 16 length.



(d) 8 length.

Figure 13: Sample goals from the `quadruped_run` task.

(a) 64 length.

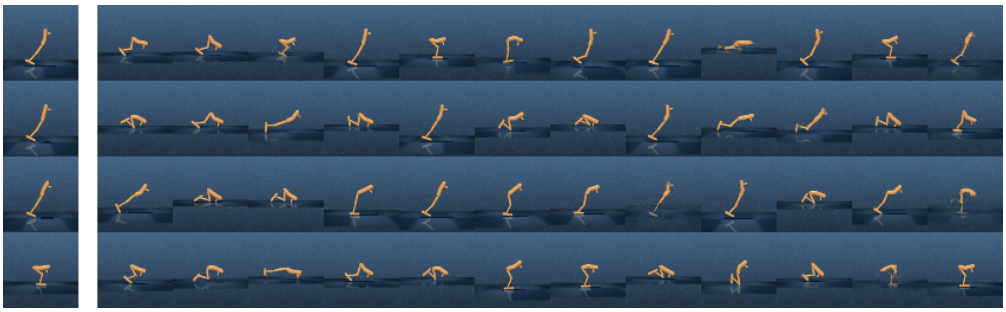


(b) 32 length.

(c) 16 length.

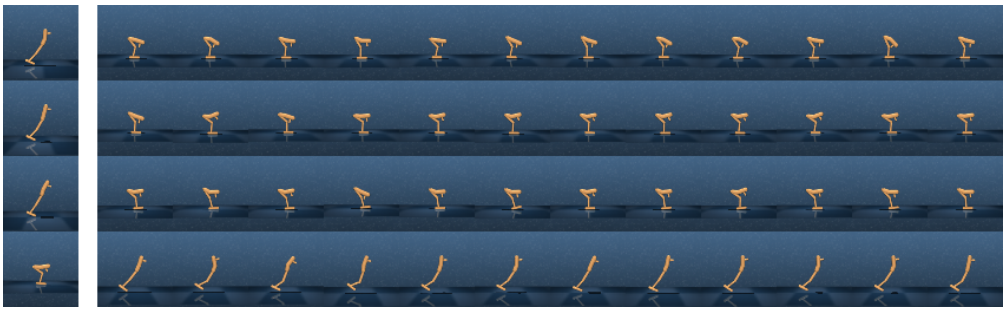
(d) 8 length.

Figure 14: Sample goals from the `cheetah_run` task.

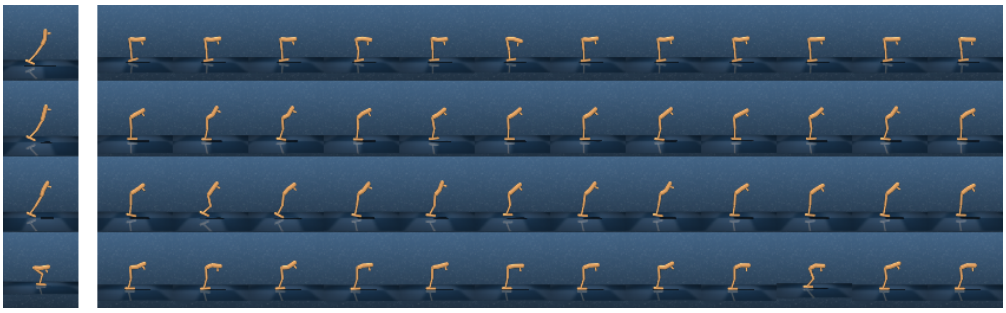


(a) 64 length.

(b) 32 length.



(c) 16 length.

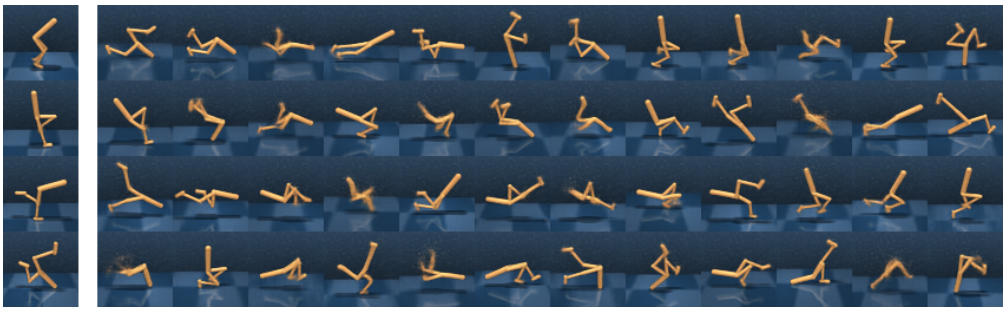


(d) 8 length.

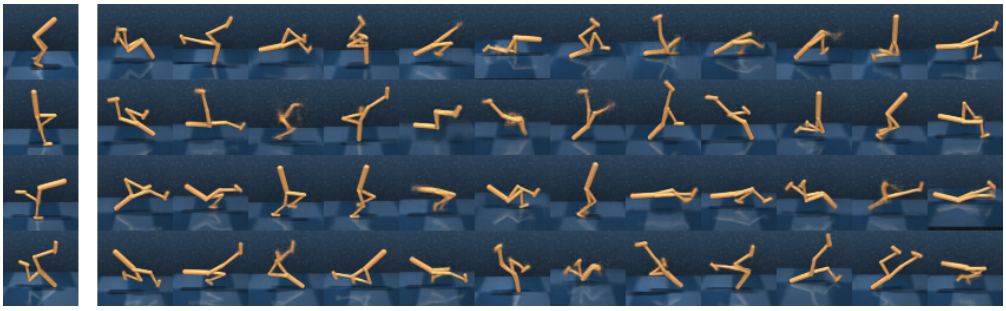
Figure 15: Sample goals from the `hopper_hop` task.

E.2 Exploration Only

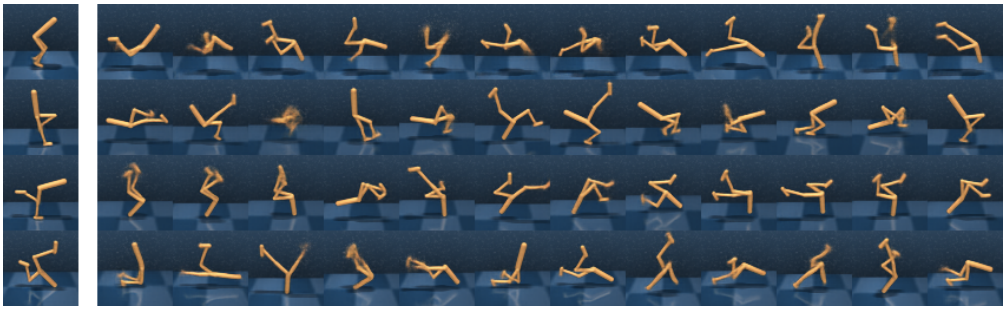
The agent is optimized solely for the exploration objective, which aims to maximize coverage of the state-transition space. We sample the goals per Skill CVAE for embodiments: `walker` (Fig. 16), `quadruped` (Fig. 17), `cheetah` (Fig. 18), and `hopper` (Fig. 19) in the DMC suite Tassa et al. (2018).



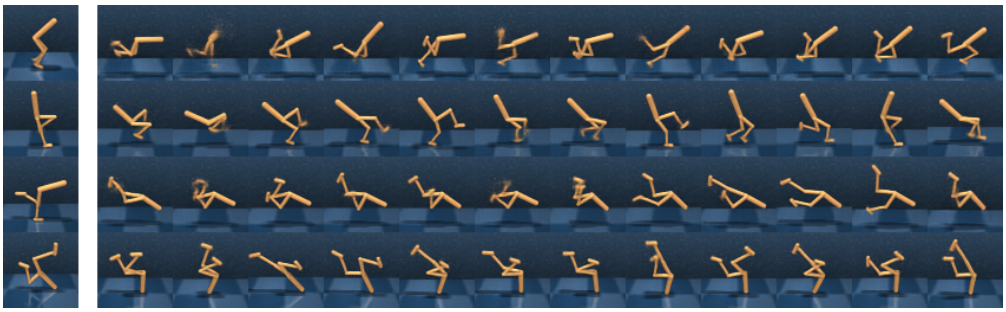
(a) 64 length.



(b) 32 length.

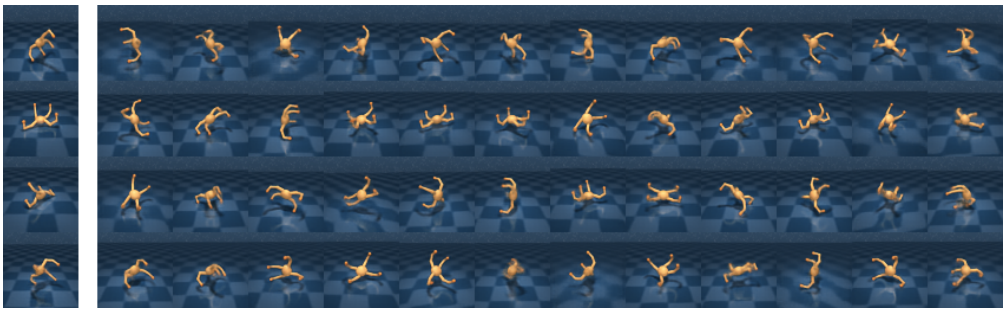


(c) 16 length.

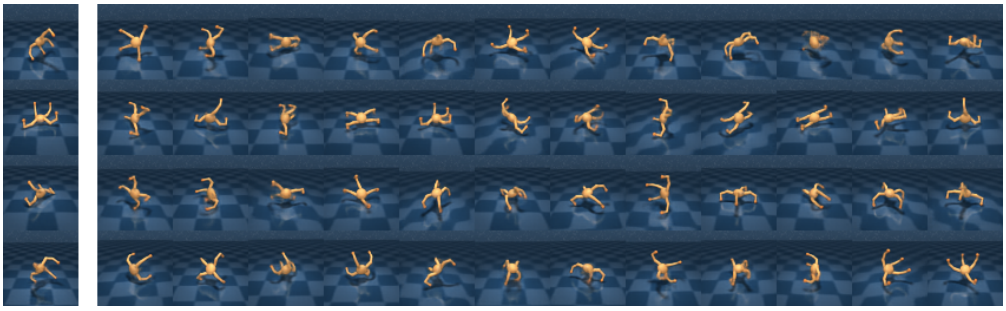


(d) 8 length.

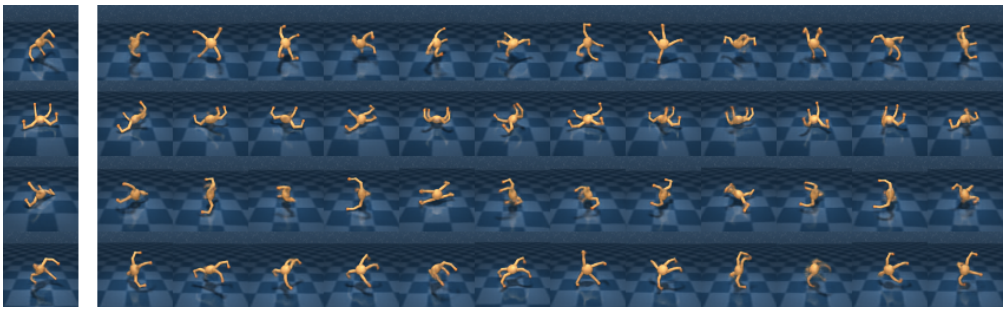
Figure 16: Sample goals learned using only the exploratory objective in a `walker` embodiment. The images on the left show the current state s_t, and the remaining images show the goal options generated by different skill CVAEs.



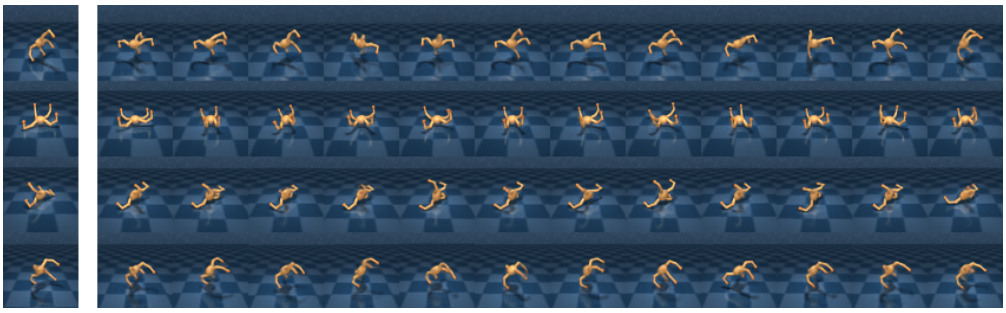
(a) 64 length.



(b) 32 length.

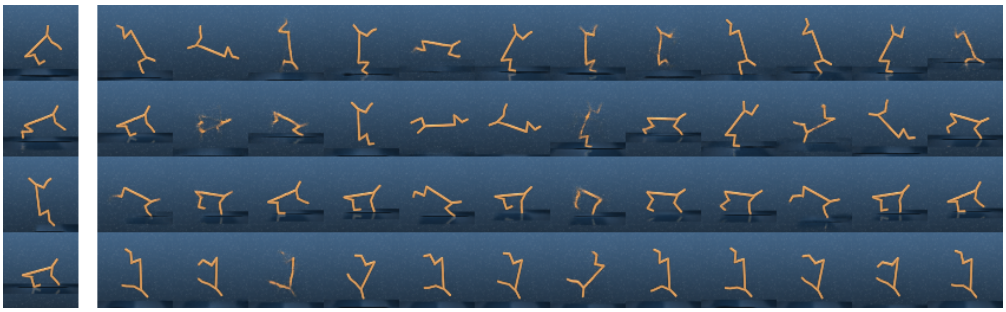


(c) 16 length.

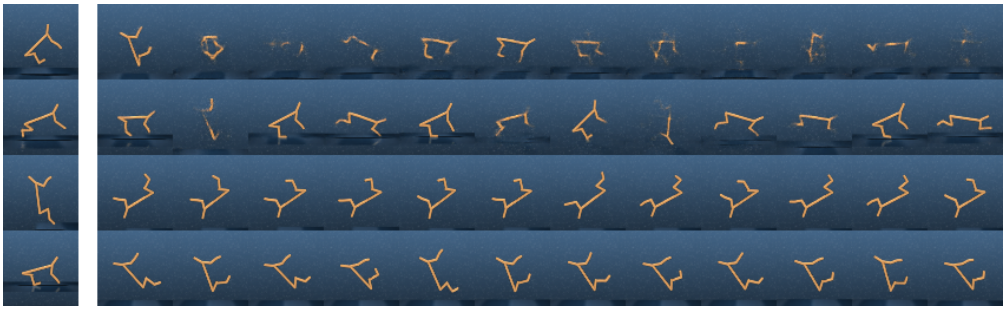


(d) 8 length.

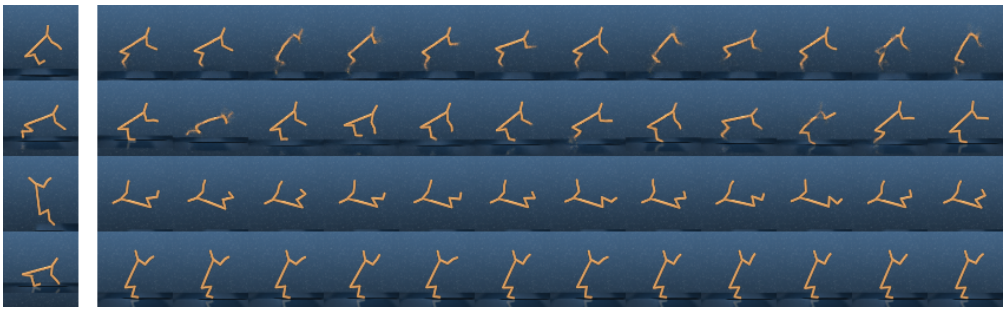
Figure 17: Sample goals from exploration as a `quadruped`.



(a) 64 length.



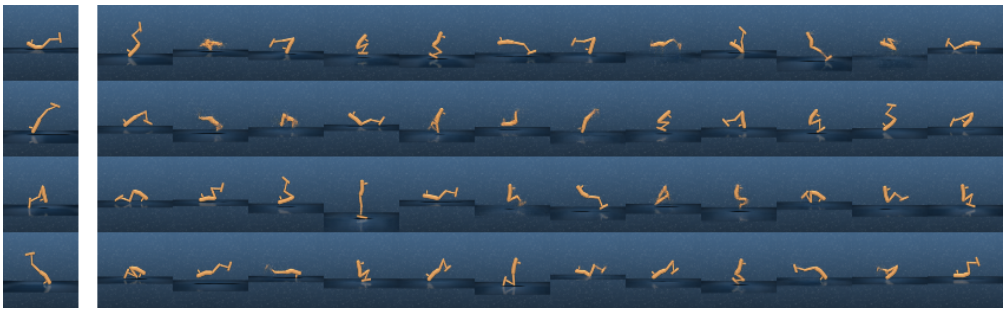
(b) 32 length.



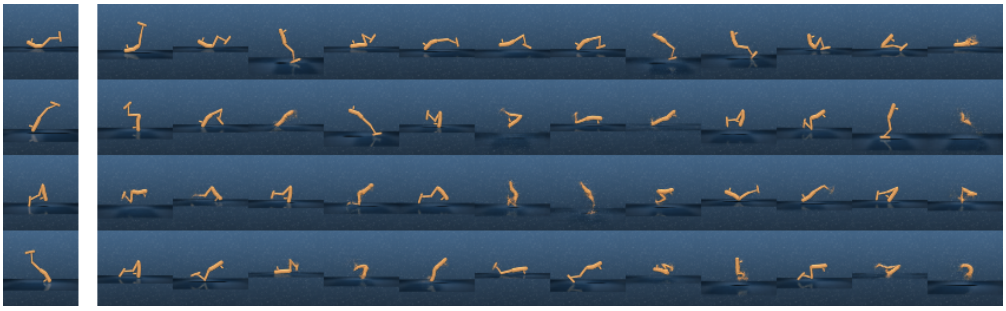
(c) 16 length.

(d) 8 length.

Figure 18: Sample goals from exploration as a `cheetah`.



(a) 64 length.



(b) 32 length.

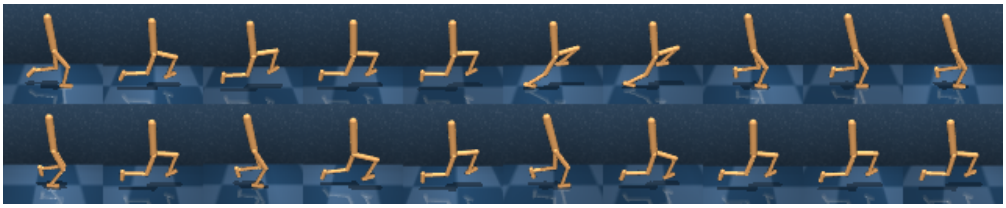
(c) 16 length.

(d) 8 length.

Figure 19: Sample goals from exploration as a `Hopper`.

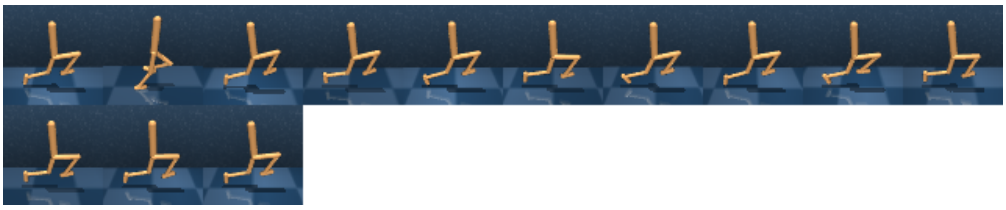
F Visualizing Choice Preferences for States

To verify if the agent exhibits any skill-length preferences in states, we run the agent for 5 episodes. Then the states s_t visited by the agent are segregated by the choice (c_t) made by the agent and shown below. The visualizations can help verify if there are any correlations between the agent state s_t and the choice variable s_t. We sample the states for the tasks: `walker_run` (Fig. 20), `quadruped_run` (Fig. 21), `cheetah_run` (Fig. 22), and `hopper_hop` (Fig. 23). A common trend observed across tasks is that inf-skills are preferred in uncommon states, such as upside-down or recovering after a misstep.

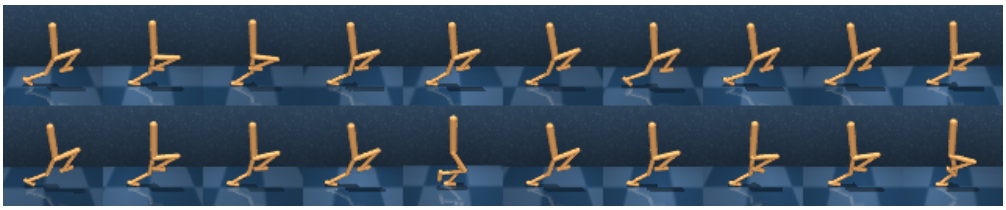


(a) 64 length.

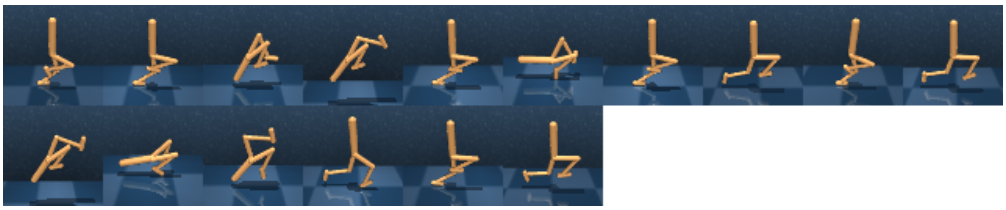
(b) 32 length.



(c) 16 length.



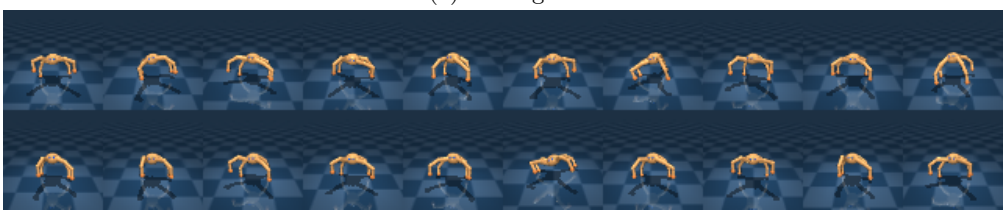
(d) 8 length.



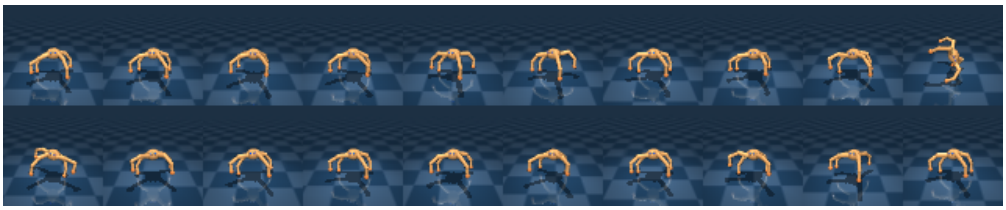
(e) ∞ length.

Figure 20: States segregated by choice for the `walker_run` task.

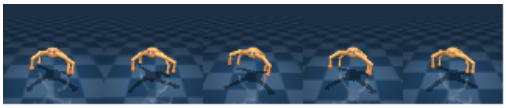
(a) 64 length.



(b) 32 length.



(c) 16 length.

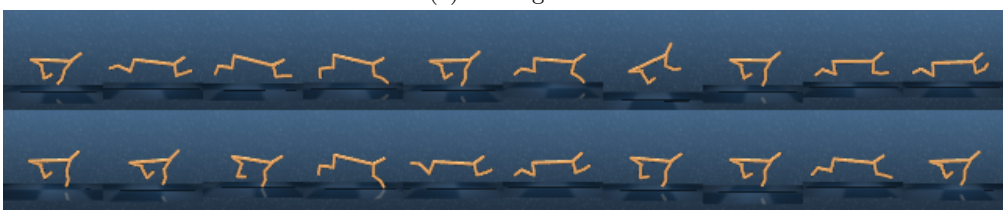


(d) 8 length.

(e) ∞ length.

Figure 21: States segregated by choice for the `quadruped_run` task.

(a) 64 length.



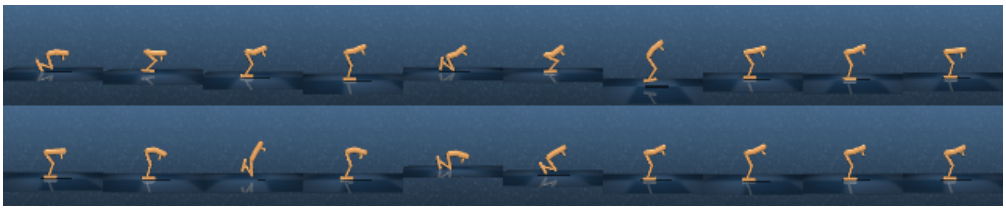
(b) 32 length.

(c) 16 length.

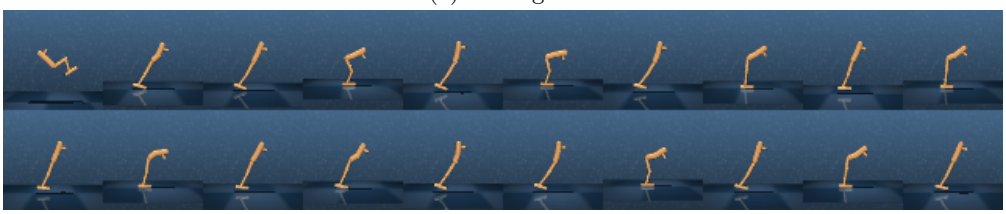
(d) 8 length.

(e) ∞ length.

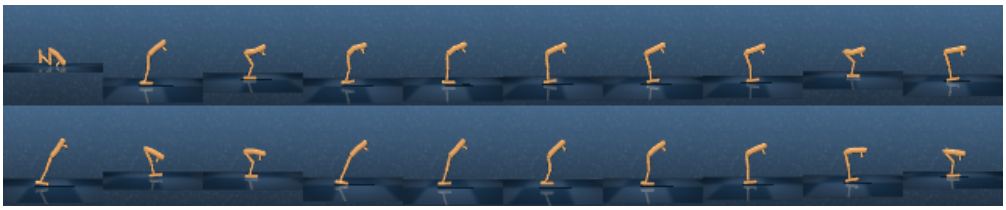
Figure 22: States segregated by choice for the `cheetah_run` task.



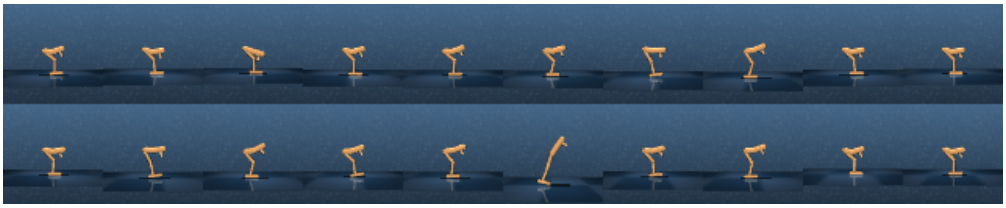
(a) 64 length.



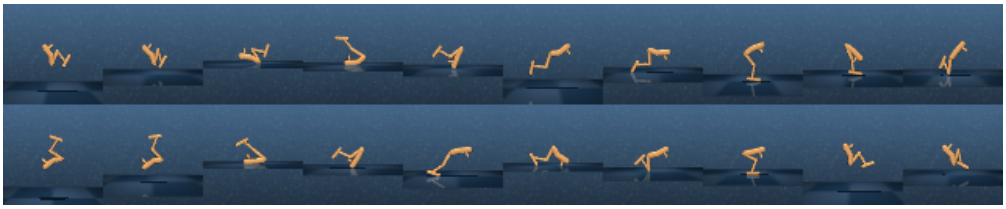
(b) 32 length.



(c) 16 length.



(d) 8 length.



(e) ∞ length.

Figure 23: States segregated by choice for the `hopper_hop` task.