Multi-Resolution Skill Discovery for Hierarchical Reinforcement Learning

Shashank Sharma
Department of Computer Science
University of Bath
ss3966@bath.ac.uk

Vinay Namboodiri
Department of Computer Science
University of Bath
vpn22@bath.ac.uk

Janina Hoffmann
Department of Psychology
University of Bath
jah253@bath.ac.uk

1 Introduction

Learning abstract actions can be beneficial for planning-based goal-conditioned reinforcement learning. Recently, offline discovery of such action or skill primitives has demonstrated effectiveness in solving long horizon tasks [1, 8]. While using abstract actions or skills has performed well, the agents usually lack finesse in motion. Humans and animals, in contrast, can learn motor skills at different levels of temporal resolution, fine-grained skills such as playing a piano, or gross skills such as running. We propose a solution to the problem of learning skills at multiple temporal resolutions to enhance control over actions. Building upon recent work, the Director, [4], we learn multiple skill encoders in parallel, one for each temporal resolution. The manager uses all skill encoders in parallel and selects from those skills using a choice variable.

We evaluate the method on a few tasks from the DeepMind control suite [12] and compare it to state-of-the-art skill-learning architectures. We find that our proposed architecture results in qualitative and quantitative performance gains in terms of convergence speed and saturation accuracy over the Director baseline. [Github: `https://github.com/shashank879/multi_skill`]

2 Related work

Hierarchical reinforcement learning (HRL) refers to a set of techniques that temporarily abstract actions [3, 13, 2, 11, 9]. While standard RL techniques like Q-Learning choose from primitive actions, HRL agents can choose temporarily extended abstract actions that differ in behavioral policies and subgoals. HRL agents are usually split into two modules, a Manager ($\mathcal{M}$) and a Worker ($\mathcal{W}$). The worker learns a task-independent policy ($\pi_{\mathcal{W}}$) that is goal-conditioned or options-based, and the manager learns a policy ($\pi_{\mathcal{M}}$) to control the worker in the context of the task. The manager works on a coarser timescale than the worker and outputs abstract actions or subgoals for the worker to complete. The manager can learn preferences over the discovered action abstractions using a model-free algorithm like the Actor-Critic [4] or use a planning mechanism for goal-conditioned reinforcement learning [8]. The temporally abstract actions help the agent plan symbolically in terms of subroutines to achieve long-term goals like reaching a specific point while avoiding obstacles. The agent simultaneously learns the abstract actions and acts by selecting these abstract actions to reach subgoals.

The abstract actions or skills are usually represented using a low-dimensional latent variable and learned by maximizing the mutual information between the trajectory (sequence of states) and the latent skill variable [1, 7, 8, 10]. The intuition behind the mutual information approach is that the trajectory is informative of the skill, and the skill is informative of the trajectory. This simultaneously

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

encourages the skills to be both diverse and predictable. OPAL [1] encodes the trajectory using a bidirectional GRU and is optimized as a Variational Autoencoder. Causal InfoGAN [7] use an InfoGAN to learn causal representations by maximizing mutual information between the skill and the initial and final state pairs. DADS [10] apply the mutual information objective to learn a diverse forward-inverse kinematics model that is later used for planning. The Director [4] is an HRL agent that outputs subgoals for the worker to complete (Sec. A). It uses a Variational Autoencoder (VAE) to remember the frequently visited states as skills and uses them to drive the worker's behavior. The preferences over the skills are learned using the Actor-Critic method. We present a novel architecture for automatic offline multi-resolution skill discovery as a modification to the Director.

3 Multi-Skill architecture

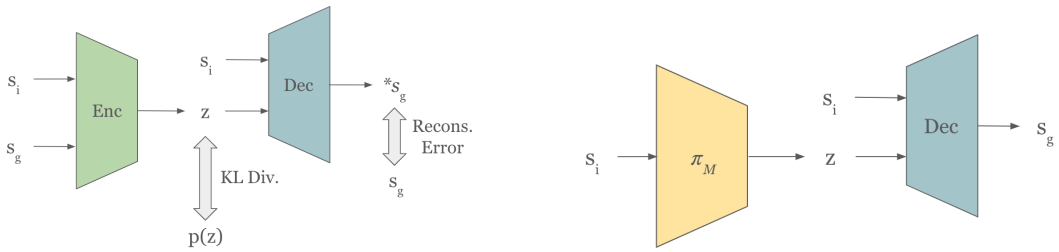
3.1 Skills as state transitions

If the manager outputs subgoals for the worker directly in the state space, it results in a high-dimensional continuous control problem for the manager [4]. For this, the Director uses a VAE to remember the most frequently visited states using a discrete VAE as *skills*. The manager outputs *skills* in the latent space, which are transformed into worker goals using the VAE decoder. In the context of a Markovian Decision Process (MDP) as a directed graph, this corresponds to learning the nodes. We propose that the agent can instead learn the frequent state transitions in the world and use them for controlling the behavior of the goal-directed worker. With the same analogy, our method corresponds to learning the edges in the directed graph. State transitions may be an efficient way of representing the world because any initial state will finally lead only to a subset of all possible states. The manager can search for the goals using edges originating from the current state. Thus, the representation inherently constrains the subgoal predictions to probable future states only, easing the manager's learning. The learned state transitions can also be seen as the modes of operation of the worker that the manager tries to learn and use while acting. Thus, we refer to the learned state transitions as skills. In implementation, our approach is similar to the forward-inverse kinematics approaches used in the past [10].

The state transitions, initial and goal state pairs (s_i, s_g), are learned using a VAE that reconstructs the goal in the context of the initial state (Fig. 1a). We use a similar distribution as the Director for the latent space (8×8), but we use a Gumble-softmax (temp $= .5$) instead of a one-hot sampling. The intuition is that we want to capture the most dominant modes in the space of state transitions. Since not all sampled state transitions will belong to distinct modes, using Gumble-softmax helps relax the fit for such cases. However, we still use one-hot distribution for the manager's policy ($\pi_{\mathcal{M}}(z|s_i)$) to sample from mode centers for most predictable state transitions. The encoder uses the initial and goal states to predict a distribution over the skills, $\text{Enc}(z|s_i, s_g)$. The decoder uses the initial state and a skill sample ($z \sim \text{Enc}(z|s_i, s_g)$) to predict the goal state, $\text{Dec}(s_i, z)$. For training, the initial and final states are sampled from an offline experience replay buffer as training pairs (s_i, s_g) to compute and optimize the loss as the ELBO objective (Eq. 3). In our case, the past experience collected by the agent is used as the buffer. Variations of the sampling strategies for (s_i, s_g) are shown in the figure 5. *Fixed-length* skills sample states with a constant temporal difference between the initial and goal states. Keeping the goal state fixed, *Variable-length* skills sample all previous states as initial states. Our initial experimentation showed that fixed-length skills offered an increase in performance but lacked finesse in motion, and the variable-length model provided much smoother motion, but the performance had high variation and failed overall due to the non-stationarity of the latent distribution.

3.2 Multi-resolution skills

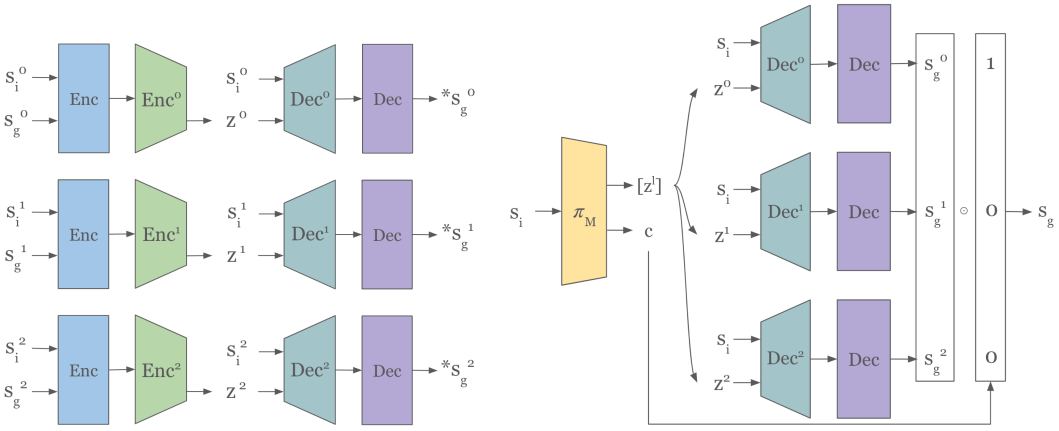
Fixed-length skills essentially allow the agent to choose states at a temporal distance k as goal states. The intuition for the multi-resolution skills is to ideally enable the agent to model any state within a specific temporal proximity, like *Variable-length* skills. Our architecture approximates this setting by learning skills at various resolutions using L separate VAEs (Fig. 2a). Each $(\text{Enc}^l, \text{Dec}^l)$ pairs in the VAEs process data at a specific temporal resolution. VAEs dedicated to various temporal resolutions allow the agent access to fine and coarse skills. All layers in the VAEs except the last encoder layer and the first decoder layer are kept shared, causing a minimal increase in model size. During training, training state pairs are extracted for each VAE separately to compute the ELBO objective $(\mathcal{L}^l)$ (Eq. 3). The total loss is computed as the sum of all losses $(\mathcal{L} = \sum_l \mathcal{L}^l)$ and is optimized in one pass.



(a) Skill VAE training architecture

(b) Acting using skill VAE

Figure 1: Illustrations of the state transition-based control for the manager. (a) Skill VAE is trained to reconstruct the goal state in the context of the initial state. (b) The manager predicts skills and then uses the skill VAE decoder to generate goals for the worker.



(a) Training architecture

(b) Acting architecture

Figure 2: Illustrations of the Multi-Skill model. (a) During training, each VAE samples training pair states (s_i^l, s_g^l) from experience replay at a different temporal resolution. The layers Enc and Dec are shared between the VAEs. (b) The manager predicts goals using the predicted skills and then selects from those goals using the choice variable.

During acting, similar to the Director, the manager produces two types of outputs every $T = 8$ steps: the skills (z^l) for each VAE separately and a one-hot choice variable (c) (Fig. 2b). The manager first predicts a set of goals using the multi-skill VAE decoders:

$$s_g^l = \text{Dec}^l(s_i, z^l) \quad \text{where} \quad z^l \sim \pi_{\mathcal{M}}(z^l|s_i), \quad l \in \{0, 1, ..., L-1\} \tag{1}$$

And then selects from those goals using a L dimensional one-hot choice variable c sampled from the choice probability distribution $(\pi_{\mathcal{M}}(c|s_i))$ predicted by the manager. Let c^l be the choice value for the l-th skill VAE, (therefore, $c^l \in \{0, 1\}$ and $\sum c^l = 1$) then the final goal is computed as:

$$s_g = \sum_l c^l \cdot s_g^l \quad \text{where} \quad c \sim \pi_{\mathcal{M}}(c|s_i) \tag{2}$$

Similar to the Director, the skill and choice are learned using the Actor-Critic method using the REINFORCE objective, but the learning signal for skills is additionally gated by the choice variable (Sec. B.2).

4 Results

We test and compare the performance of the Multi-skill agent against the results from the Director at a few tasks from the Deepmind Control suite (Fig. 3). The tasks chosen are: `walker_walk`,

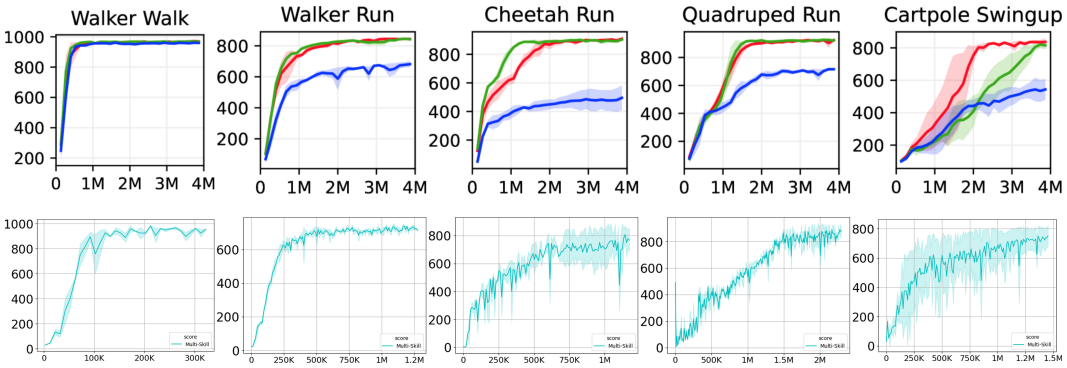


Figure 3: Performance scores for: — Director, — Director with worker external task reward, — DreamerV2 and — Multi-Skill agent. (See Fig. 6 for approximate projections).

`walker_run`, `cheetah_run`, `quadruped_run` and `cartpole_swingup` as we are primarily interested in locomotion. The skill resolutions for the Multi-skill agent are set to $[64, 32, 16]$ (Fig. 5).

It can be seen that the proposed model shows improvements over the default Director and even compares to the DreamerV2 [6]. The authors of the Director mention its high sample efficiency when comparing it to the DreamerV2. Our model maintains the sample efficiency of the Director while improving performance. Also, we do not optimize the worker for the external task rewards, thus preserving its generality and potential for transfer learning across tasks. Upon qualitative analysis, the multi-skills Agent motion was smoother with reduced 'jitters' and fewer cases of imbalance. The agent also showed evidence of possessing fine and coarse skills (Fig. 10), where it precisely balances itself in an unstable pose and later walks/runs quickly. We also measure the quality of the generated goals using the reconstruction error between the actual and predicted final states using the model (Sec. B.1). The reconstruction error for the Multi-Skill agent was relatively lower than for the Director (Fig. 9). Finally, a histogram showing the distribution of the choice for skill VAE is shown in figure 7.

5 Discussion

It should be noted that while the manager generates goals at different temporal resolutions, it can choose a new goal every $T = 8$ steps. If the worker never completes the goals, it naturally raises questions about the requirement of higher temporal resolution skills. To answer this question, we first observe the subgoal achievement rate in the default Director (Fig. 8). It can be seen that the worker rarely completes the assigned goals, indicating that without constraints, the agent prefers not to complete its subgoals. To induce subgoal completion, we experiment using only $k = 8$ length skills, which results in complete failure. The observations lead to the conclusion that, at least in continuous domains, the worker subgoal is not meant to be completed and instead serves as a distant carrot that the worker constantly follows. Within that analogy, the Multi-skill model allows dangling the carrot at different distances. The effect of longer-distance subgoals can be seen in the smooth brush tool in painting software, where the mouse cursor (goal carrot) pulls the brush (worker) with a fixed-length string. A longer string produces a smoother but less precise stroke, and a shorter string produces a coarser but more precise stroke. The Multi-skill model serves the same analogy and provides the agent with skills at different temporal resolutions, accessed by the choice variable, allowing fine and gross control over motion.

6 Conclusion & Future work

We propose a method to learn skills at multiple temporal resolutions, resulting in significant performance improvement and some qualitative dexterity in motion. The architecture is scalable and allows adding multiple skill VAEs with minimal increments in model size. In the future, we plan to experiment with contextual and task/agent-specific sampling strategies for (s_i, s_g). The learned abstractions can also be used as elements in a planning mechanism for a goal-conditioned RL where the reconstruction error using the skill VAE can provide a reachability measure.

References

[1] Anurag Ajay et al. "{OPAL}: Offline Primitive Discovery for Accelerating Offline Reinforcement Learning". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=V69LGwJ0lIN.

[2] Andrew G Barto and Sridhar Mahadevan. "Recent advances in hierarchical reinforcement learning". In: *Discrete event dynamic systems* 13.1 (2003), pp. 41–77.

[3] Matthew M Botvinick, Yael Niv, and Andew G Barto. "Hierarchically organized behavior and its neural foundations: A reinforcement learning perspective". In: *Cognition* 113.3 (2009), pp. 262–280.

[4] Danijar Hafner et al. "Deep Hierarchical Planning from Pixels". In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo et al. Vol. 35. Curran Associates, Inc., 2022, pp. 26091–26104. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/a766f56d2da42cae20b5652970ec04ef-Paper-Conference.pdf.

[5] Danijar Hafner et al. "Learning latent dynamics for planning from pixels". In: *International conference on machine learning*. PMLR. 2019, pp. 2555–2565.

[6] Danijar Hafner et al. "Mastering atari with discrete world models". In: *arXiv preprint arXiv:2010.02193* (2020).

[7] Thanard Kurutach et al. "Learning plannable representations with causal infogan". In: *Advances in Neural Information Processing Systems* 31 (2018).

[8] Jinning Li et al. "Hierarchical Planning Through Goal-Conditioned Offline Reinforcement Learning". In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 10216–10223. DOI: 10.1109/LRA.2022.3190100.

[9] Shubham Pateria et al. "Hierarchical Reinforcement Learning: A Comprehensive Survey". In: *ACM Comput. Surv.* 54.5 (June 2021). ISSN: 0360-0300. DOI: 10.1145/3453160. URL: https://doi.org/10.1145/3453160.

[10] Archit Sharma et al. "Dynamics-Aware Unsupervised Discovery of Skills". In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=HJgLZR4KvH.

[11] Richard S Sutton, Doina Precup, and Satinder Singh. "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning". In: *Artificial intelligence* 112.1-2 (1999), pp. 181–211.

[12] Yuval Tassa et al. "Deepmind control suite". In: *arXiv preprint arXiv:1801.00690* (2018).

[13] Marco A Wiering and Martijn Van Otterlo. "Reinforcement learning". In: *Adaptation, learning, and optimization* 12.3 (2012), p. 729.

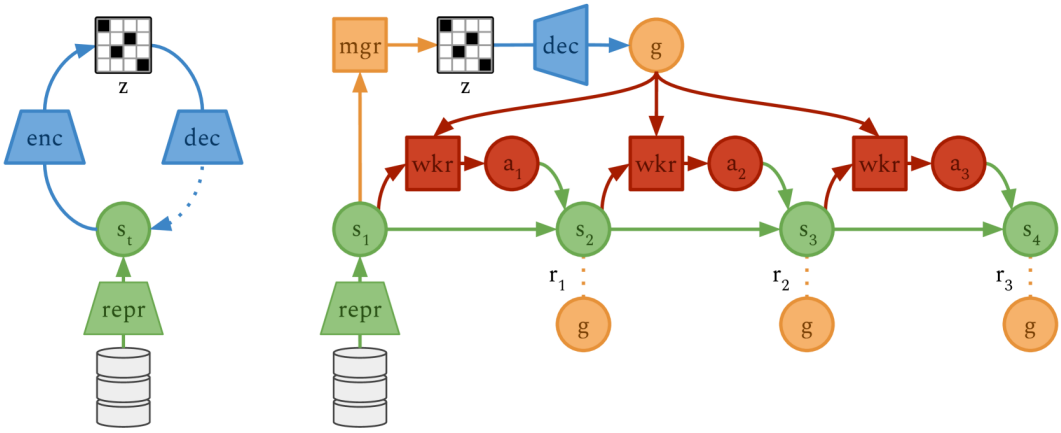


Figure 4: Courtesy of [4]. Illustration of the Director. [Left] The goal autoencoder compresses the state vectors into vectors of discrete codes. [Right] The manager selects abstract actions in this discrete space every $T = 8$ steps, which the goal decoder then turns into feature space goals. The worker takes the current and goal state as input to output primitive actions to maximize similarity rewards.

A Director

Director [4] is one of the recent state-of-the-art methods in HRL (Fig. 4). The agent takes the current observation (o_i) as input from the environment, processed using a temporally-extended perception layer (RSSM [5]) that integrates a state representation ($s_i \in \mathbb{S}$) using observations over time. It then learns VAE that compresses the state to a low-dimensional categorical variable (z), essentially allowing it to remember the top n most visited states. The manager takes the current state (s_i) as input to select goals ($s_g \in \mathbb{S}$) using the latent variable (z) for the worker (every $T = 8$ steps), and the worker tries to reach those goals. The manager and the worker are optimized using the Actor-Critic algorithm for different rewards. For the manager, it is the external task and exploration rewards; for the worker, it is the similarity measure between the goal and the final reached state. Our model is presented as a modification to the goal-generation method of the Director.

B Equations

B.1 Skill VAE objective

Given a training example, a pair of initial and final states as (s_i, s_g), the ELBO objective for a single skill VAE is:

$$\mathcal{L}(\phi) = \|s_g - \text{Dec}_\phi(s_i, z)\|^2 + \beta \text{KL}[\text{Enc}_\phi(z|s_i, s_g) \parallel p(z)] \quad \text{where} \quad z \sim \text{Enc}_\phi(z|s_i, s_g) \tag{3}$$

In the case of Multi-resolution skills, the Enc_ϕ and Dec_ϕ consist of the shared and the level-specific layers. The state is a continuous 1024-dimensional variable, and the skill is a set of 8, 8 Gumbel-softmax distribution samples (i.e., 8×8). For this, the encoder outputs an 8×8 matrix of logits, and a Gumbel-softmax distribution (temperature = .5) is applied on the last axis. However, the manager also outputs in the same skill space but uses one-hot categorical distributions for sampling skills from the policy ($\pi_\mathcal{M}(z|s_i)$).

It can be observed that when the distance between the states is very high, the final state can no longer be explained by the initial state. At that point, the model regresses to the model proposed in the Director, i.e., predicts the final state regardless of the initial state. So, the skills only carry information about the goal state, not the state transition.

The quality of the learned goal-generating model is measured using the reconstruction error for the goal states. For Director, it is $\|s - \text{Dec}_\phi(z)\|^2$ where $z \sim \text{Enc}_\phi(z|s)$. And for the Multi-skill model, it is $\min_l \|s_g - \text{Dec}^l_\phi(s_i, z)\|^2$ where $z \sim \text{Enc}^l_\phi(z|s_i, s_g)$ and l is VAE index.

B.2 Manager Actor-Critic objective

The policy is optimized using a similar Actor-critic-based approach used for the Director. The agent collects on-policy experience using imagined rollouts in the latent space using the RSSM module. The imagined trajectories are used to compute the λ-returns at all time steps. The policy is then optimized for the REINFORCE objective, with entropy added as an additional loss. In the case of the Multi-skill model, the same objective is calculated for both manager outputs, choice, and skill. However, we removed the entropic loss for the choice as it seemed to induce slightly slow learning. The choice is optimized using the default REINFORCE objective 4. But the REINFORCE skill objective is additionally gated by the choice 5. In other words, the agent only learns about the skills it chooses.

$$\mathcal{L}(\pi_c) = -\mathbb{E}_{\pi_c, p_\phi}\left[\sum_{t=1}^{H} \ln \pi_c(c|s_t)\mathrm{sg}(V_t^\lambda - v(s_t))\right] \tag{4}$$

$$\mathcal{L}(\pi_z^l) = -\mathbb{E}_{\pi_z^l, p_\phi}\left[\sum_{t=1}^{H} \ln \pi_z^l(z^l|s_t)c^l\mathrm{sg}(V_t^\lambda - v(s_t)) + \eta \mathrm{H}[\pi_z^l(z^l|s_t)]\right] \tag{5}$$

C Additional Images

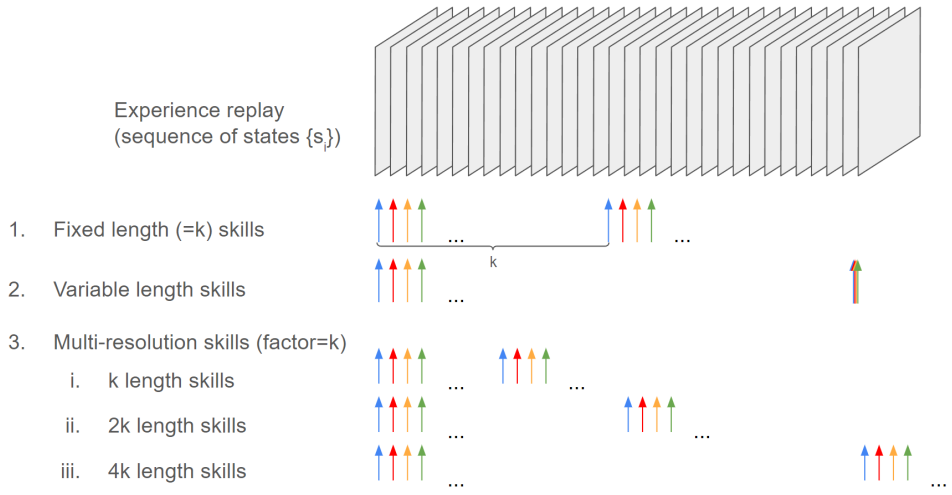


Figure 5: Illustration of the initial and goal state sampling process for the variants. The same color arrows indicate the initial and final state pairs (s_i, s_g).

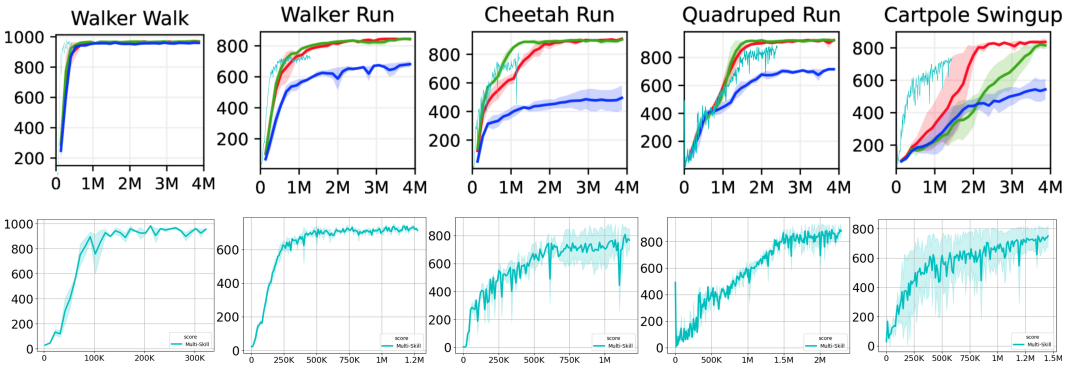


Figure 6: Approximate projections of the Multi-Skill results on the original results from the Director.

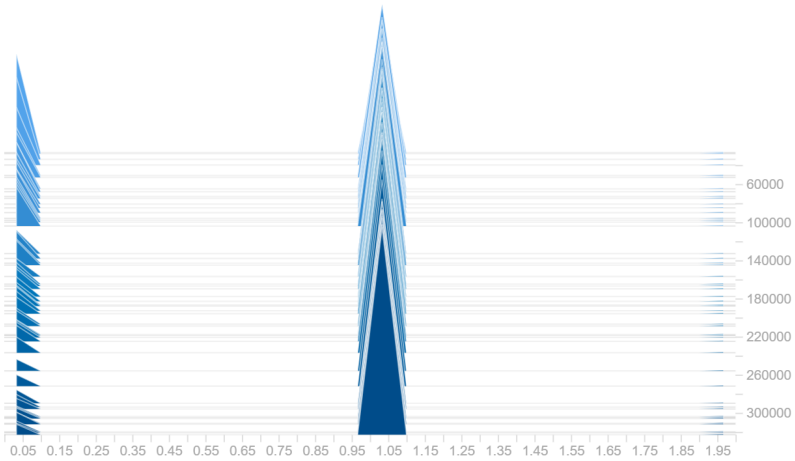


Figure 7: Categorical histogram of the skill choice made by the Multi-skill agent over the course of the training at the task of `walker_walk`. The indices $0, 1, 2$ correspond to skill lengths $k = 64, 32, 16$ respectively.

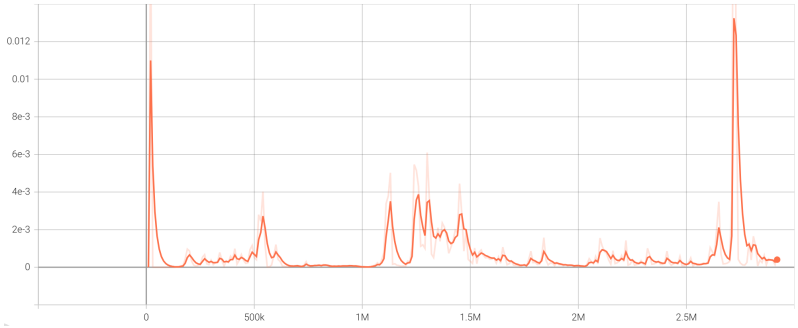


Figure 8: The average rate of subgoal completion by the worker in the default Director. [Note the scale of the y-axis].

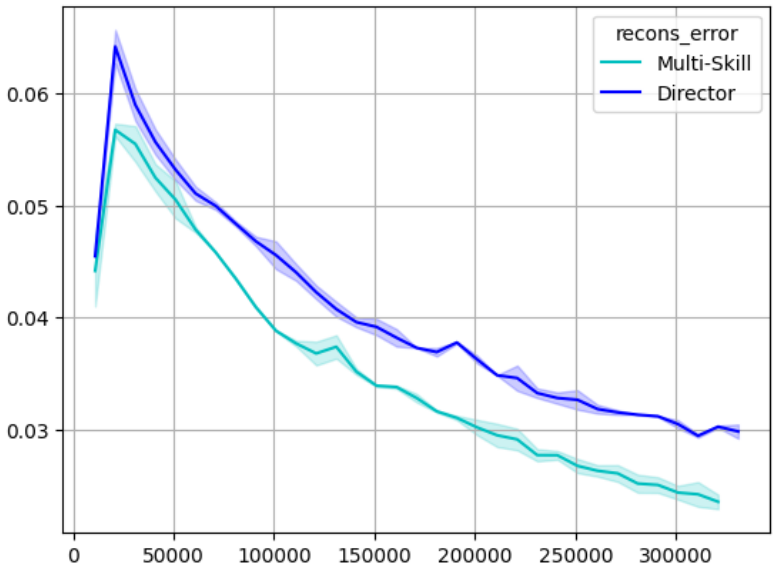


Figure 9: Plot of the average goal reconstruction errors at the task of `walker_walk`.

D Tasks

We evaluate our model at the tasks of `walker_walk`, `walker_run`, `cheetah_walk`, `quadruped_run` and `cartpole_swingup` in Deepmind Control suite [12]. The tasks involve moving an agent in a physics-based environment. Since the objects in the environment have momentum, the agent is required to have dexterous movements. We use `walker_walk` as a baseline where the Director performs as well as other methods. And we pick four other tasks where the Director has a margin for improvement against other methods.

E Performance Samples

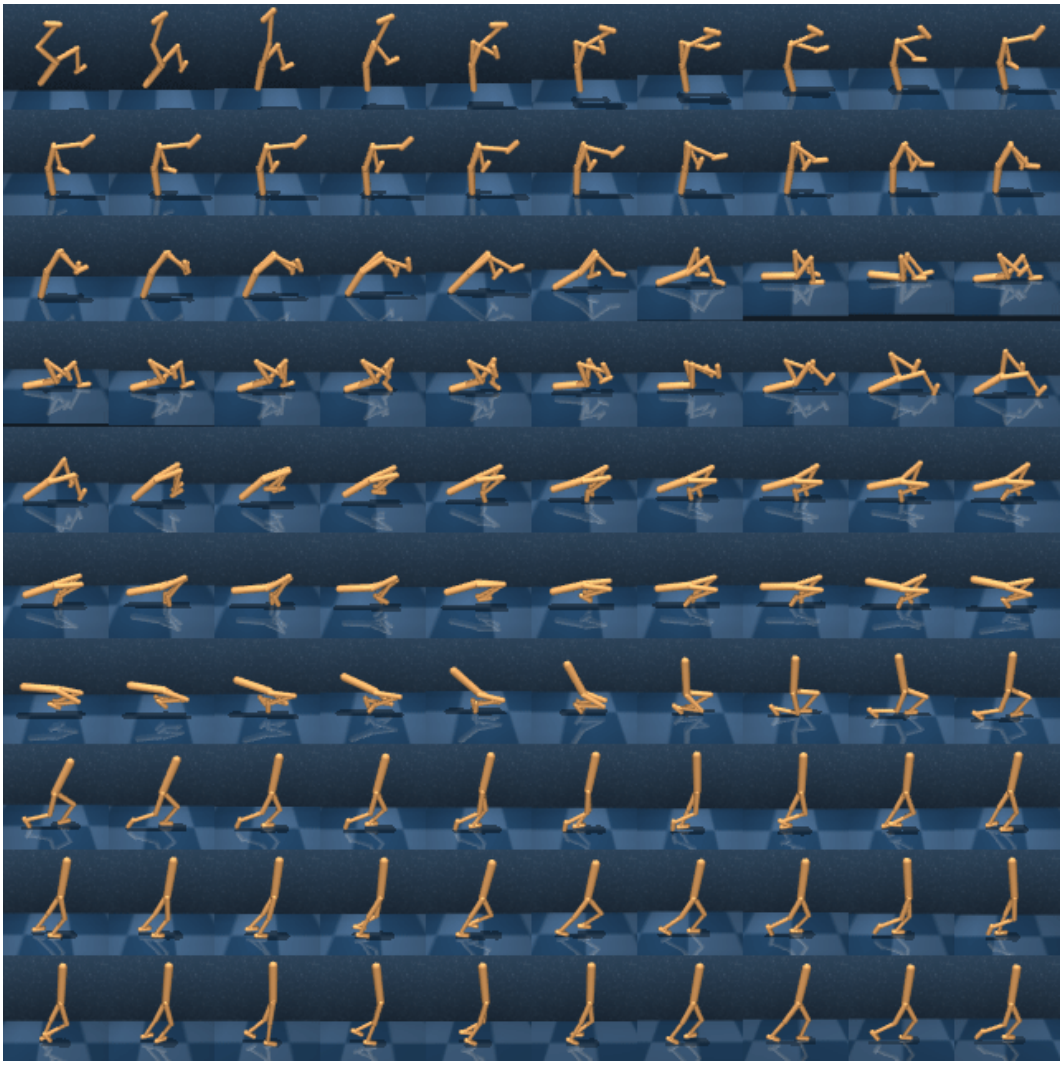


Figure 10: Frames (every second frame) from the performance of the Multi-resolution skill model after 120k steps of training at the task of `walker_walk` (first sample after reaching saturation). The agent successfully uses its fine motor skills to balance itself on its toes and then launches itself upwards to start walking forward using the coarser resolution skills. The motion is smooth and suffers fewer cases of imbalances in an episode.

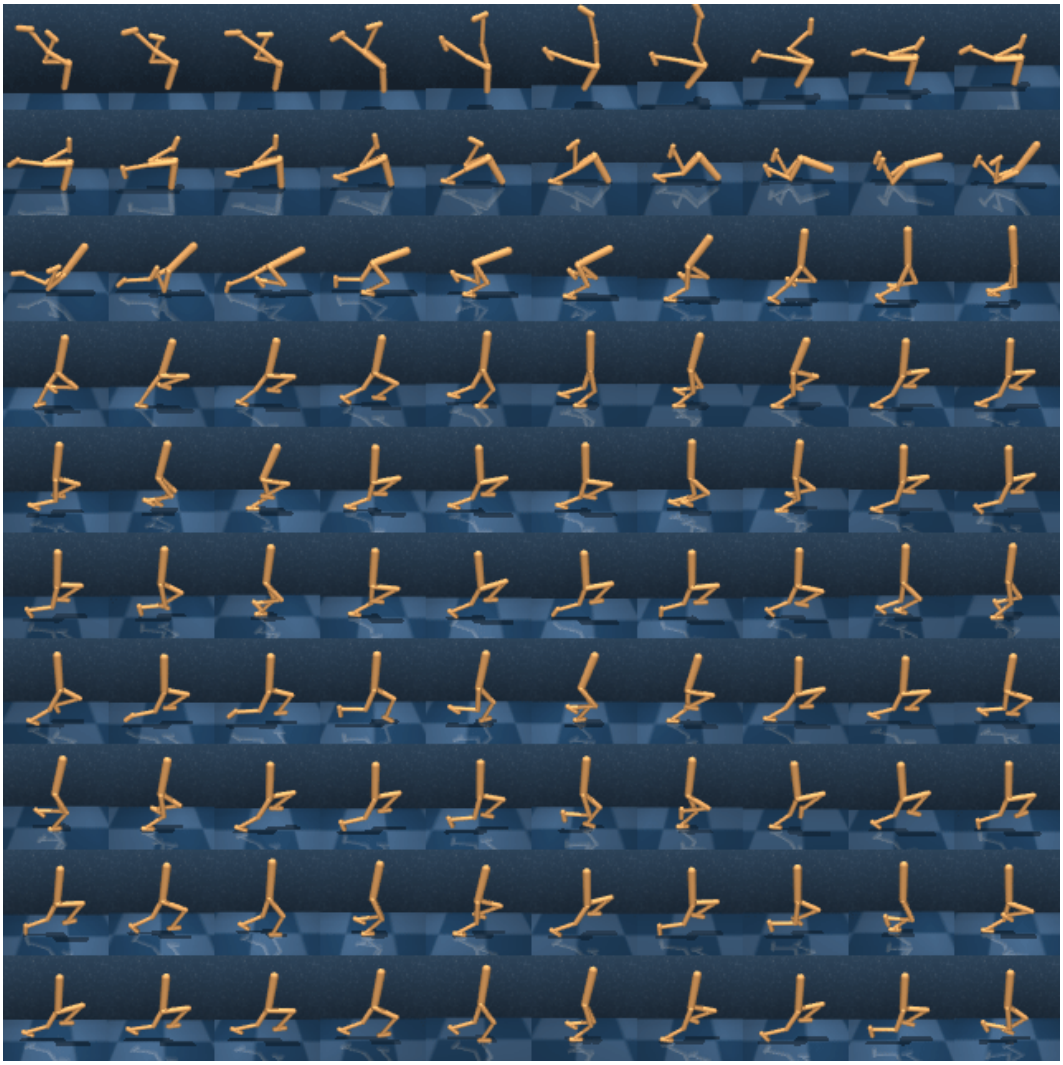


Figure 11: Frames (every fourth frame) from the final performance of the Multi-resolution skill model at the task of `walker_run`.

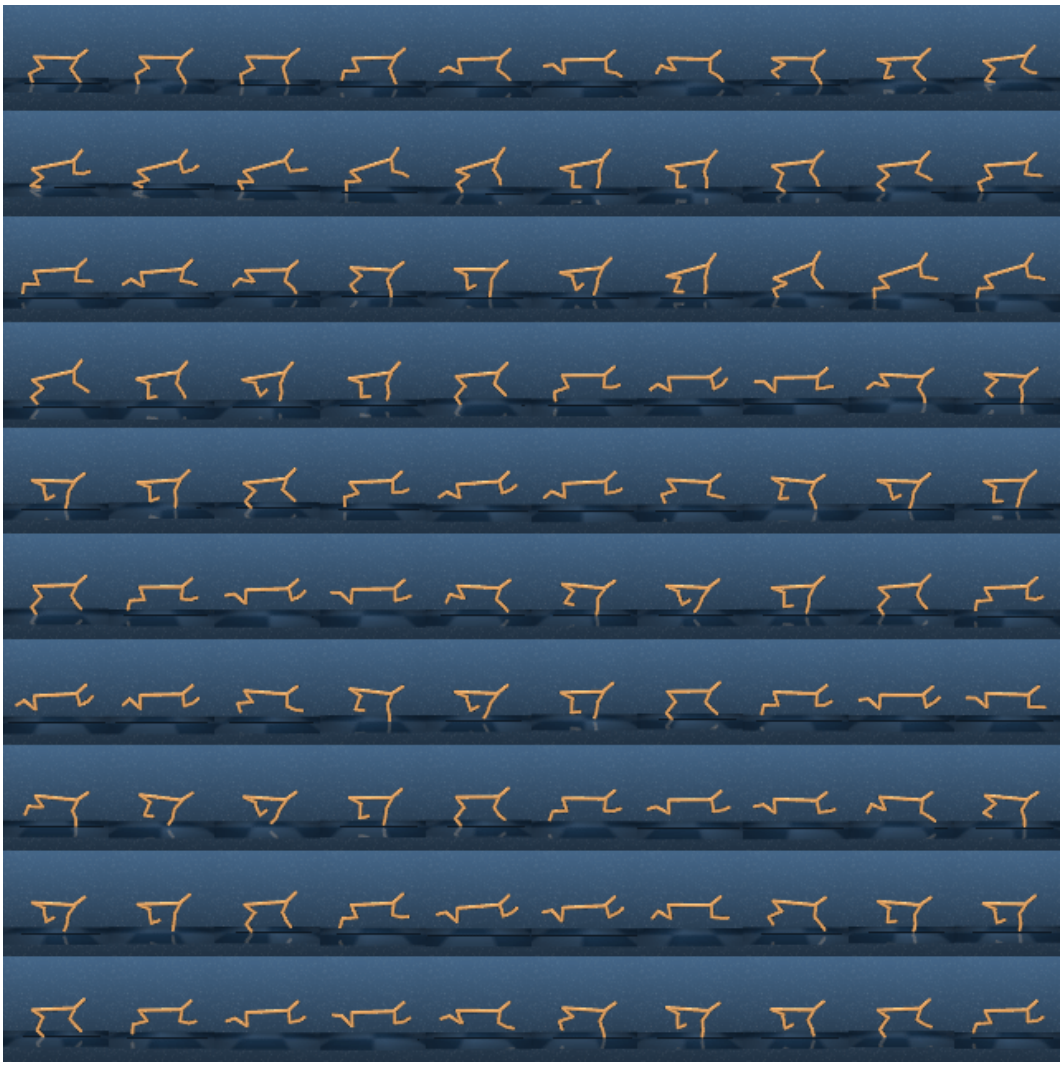


Figure 12: Frames (every fourth frame) from the final performance of the Multi-resolution skill model at the task of `cheetah_run`. The model automatically learns the rhythm of alternate pushing with the front and hind legs. Leading to efficient and consistent forward thrust.

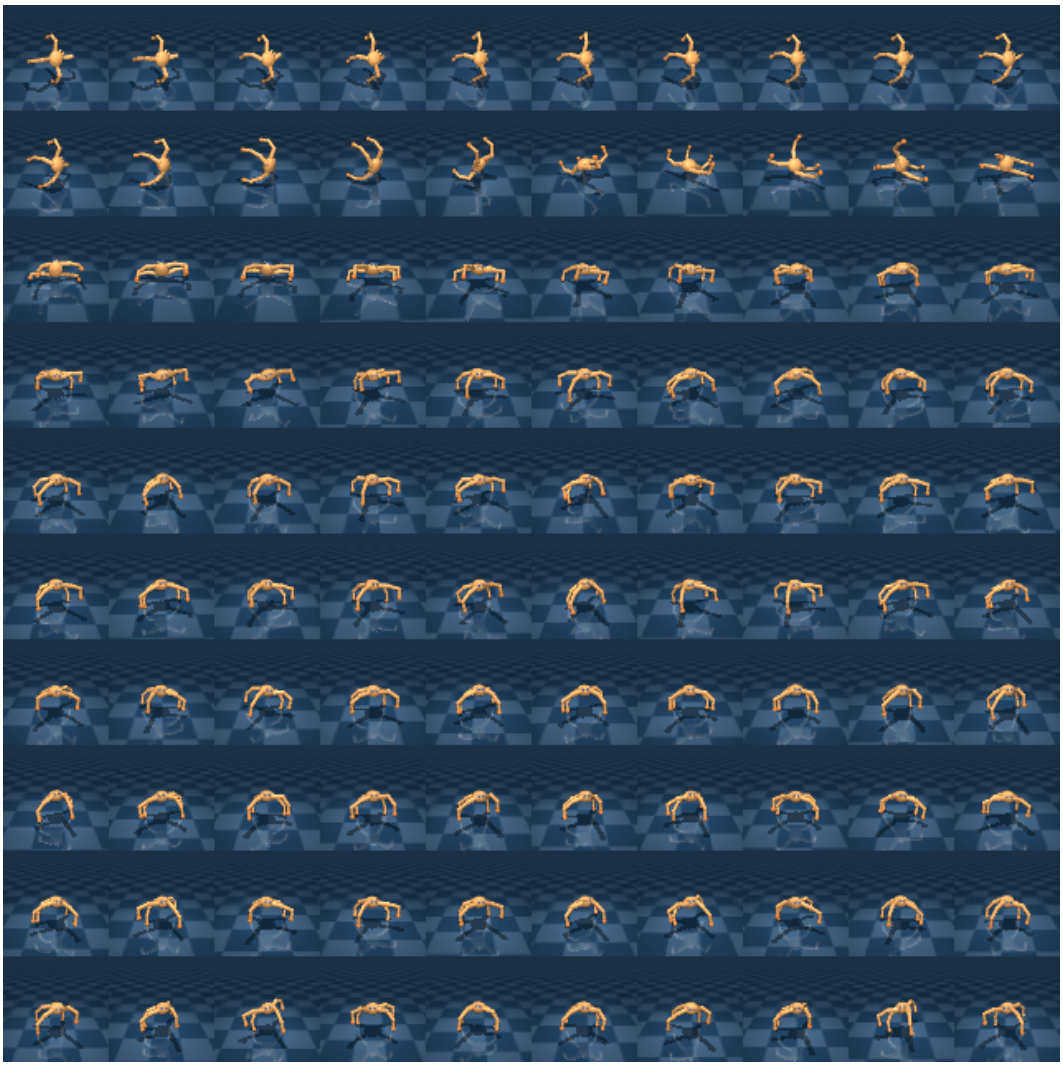


Figure 13: Frames (every fourth frame) from the final performance of the Multi-resolution skill model at the task of `quadruped_run`. The model mostly uses the hind legs for walking to maintain a higher height for greater rewards.

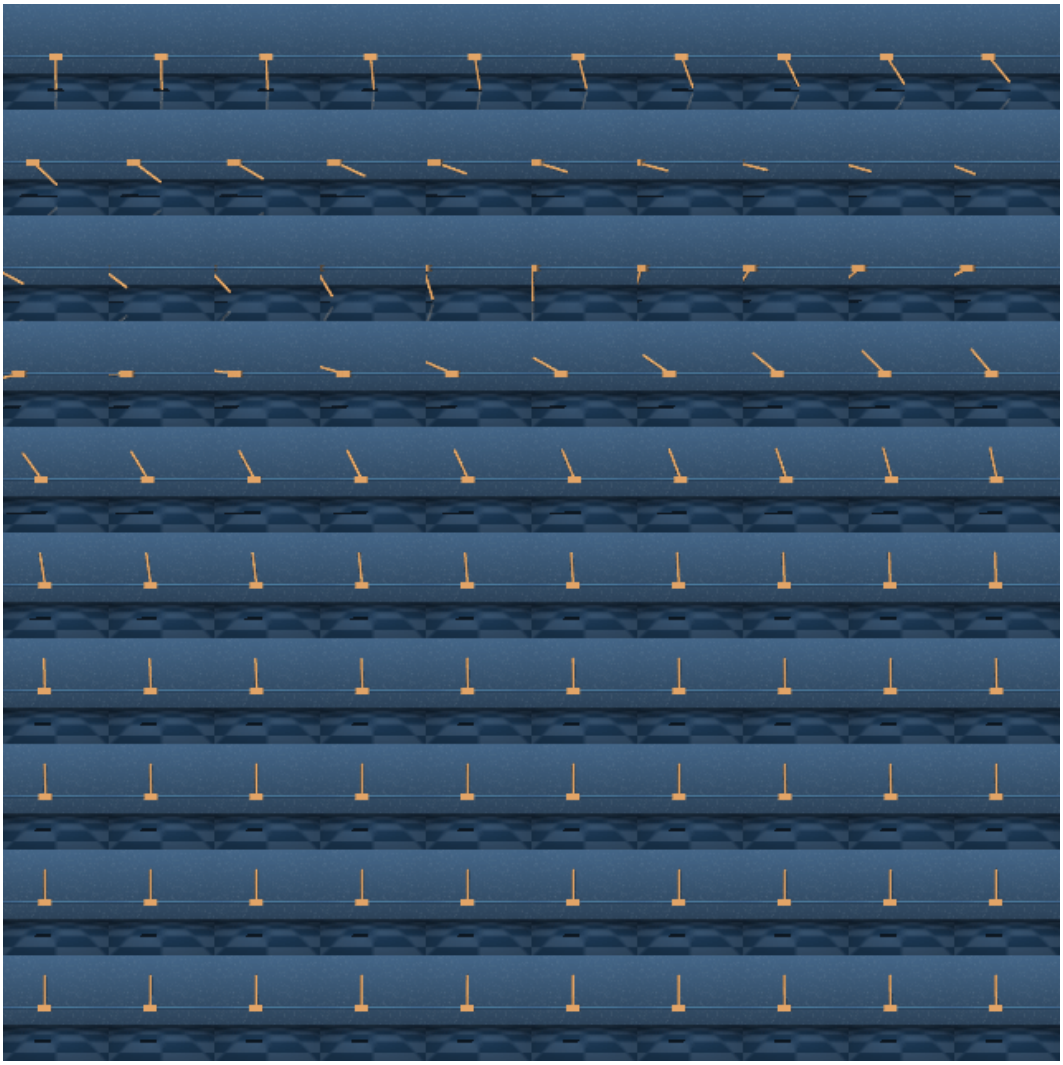


Figure 14: Frames (every fourth frame) from the final performance of the Multi-resolution skill model at the task of `cartpole_swingup`.