Supplementary Material

A. Additional Experiments

A.1. Decomposition and Option Clustering

We further deploy HO2 on a set of simple locomotion tasks, where the goal is for an agent to move to one of three randomized target locations in a square room. These are specified as a set of target locations and a task index to select the target of interest.

The main research questions we aim to answer (both qualitatively and quantitatively) are: (1) How do the discovered options specialize and represent different behaviors?; and (2) How is this decomposition affected by variations in the task, embodiment, or algorithmic properties of the agent? To answer these questions, we investigate a number of variations:

- Three bodies: a quadruped with two ("Ant") or three ("Quad") torque-controlled joints on each leg, and a rolling ball ("Ball") controlled by applying yaw and forward roll torques.
- With or without *information asymmetry* (IA) between high- and low-level controllers, where the task index and target positions are withheld from the options and only provided to the categorical option controller.
- With or without a limited number of switches in the optimization.

Information-asymmetry (IA) in particular, has recently been shown to be effective for learning general skills (Galashov et al., 2018): by withholding task-information from low-level options, they can learn task-agnostic, temporally-consistent behaviors that can be composed by the option controller to solve a task. This mirrors the setup in the aforementioned Sawyer tasks, where the task information is only fed to the high-level controller.

For each of the different cases, we qualitatively evaluate the trained agent over 100 episodes, and generate histograms over the different options used, and scatter plots to indicate how options cluster the state/action spaces and task information. We also present quantitative measures (over 5 seeds) to accompany these plots, in the form of (1) Silhouette score, a measure of clustering accuracy based on inter- and intra-cluster distances[2]; and (2) entropy over the option histogram, to quantify diversity. The quantitative results are shown in Table 2, and the qualitative plots are shown in Figure 10. Details and images of the environment are in Section B.4.

[2]The silhouette score is a value in $[-1, 1]$ with higher values indicating cluster separability. We note that the values obtained in this setting do not correspond to high *absolute* separability, as multiple options can be used to model the same skill or behavior abstraction. We are instead interested in the *relative* clustering score for different scenarios.

The results show a number of trends. Firstly, the usage of IA leads to a greater diversity of options used, across all bodies. Secondly, with IA, the options tend to lead to specialized actions, as demonstrated by the clearer option separation in action space. In the case of the $2D$ action space of the ball, the options correspond to turning left or right (y-axis) at different forward torques (x-axis). Thirdly, while the simple Ball can learn these high-level body-agnostic behaviors, the options for more complex bodies have greater switch rates that suggest the learned behaviors may be related to lower-level motor behaviors over a shorter timescale. Lastly, limiting the number of switches during marginalization does indeed lead to a lower switch rate between options, without hampering the ability of the agent to complete the task.

A.2. Action and Temporal Abstraction Experiments

The complete results for all pixel and proprioception based multitask experiments for ball-in-cup and stacking can be respectively found in Figures 11 and 12. Both RHPO and HO2 outperform a simple Gaussian policy trained via MPO. HO2 additionally improves performance over mixture policies (RHPO) demonstrating that the ability to learn temporal abstraction proves beneficial in these domains. The difference grows as the task complexity increases and is particularly pronounced for the final stacking tasks.

A.3. Task-agnostic Terminations

The perspective of options as task-independent skills with termination conditions as being part of a skill, leads to termination conditions which are also task independent. We show that at least in this limited set of experiments, the perspective of task-dependent termination conditions - i.e. with access to task information - which can be understood as part of the high-level control mechanism for activating options improves performance. Intuitively, by removing task information from the termination conditions, we constrain the space of solutions which first accelerates training slightly but limits final performance. It additionally shows that while we benefit when sharing options across tasks, each task gains from controlling the length of these options independently. Based on these results, the termination conditions across all other multi-task experiments are conditioned on the active task.

The complete results for all experiments with task-agnostic terminations can be found in Figure 13.

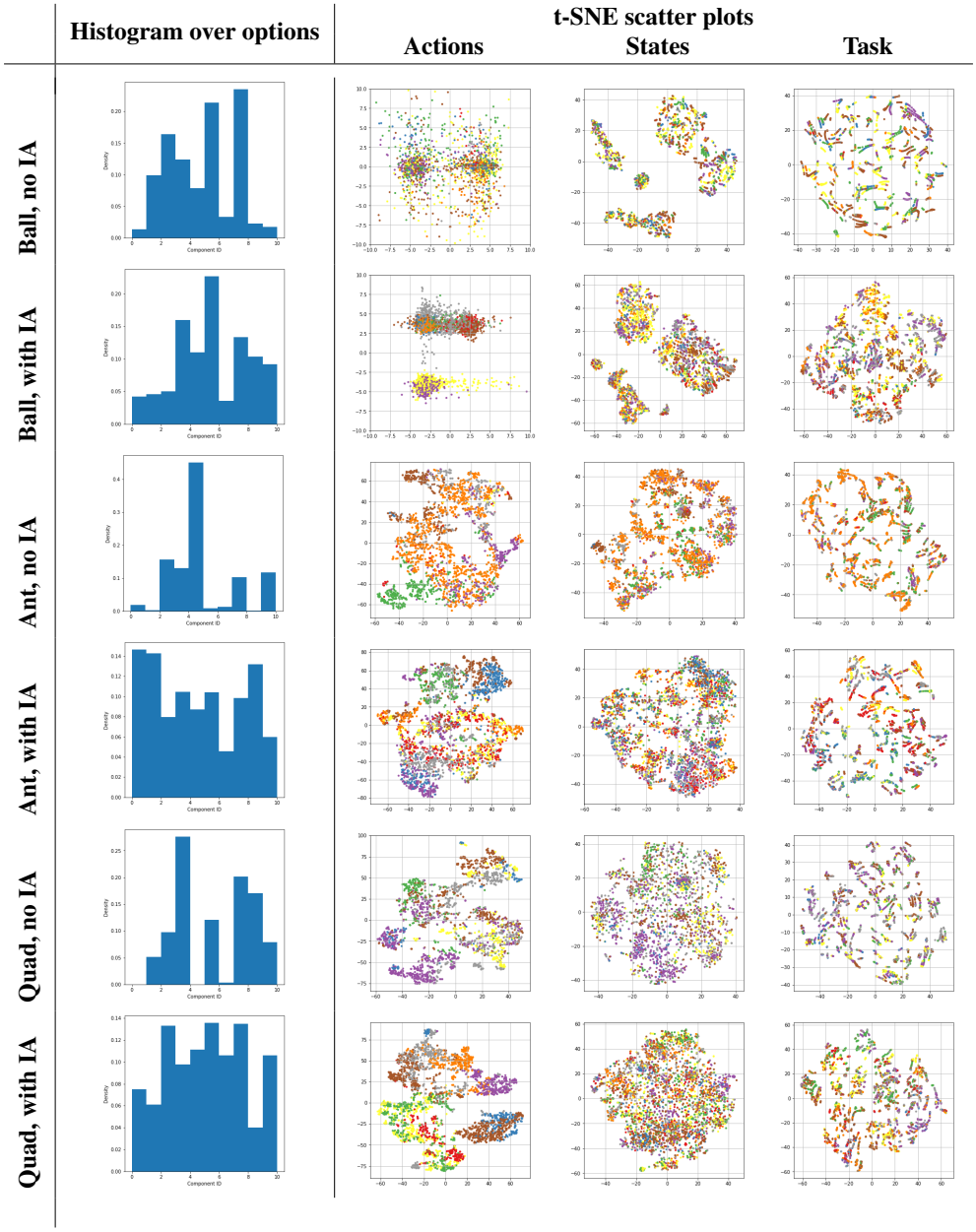

Figure 10: Qualitative results for the three bodies (Ball, Ant, Quad) without limited switches, both with and without IA, obtained over 100 evaluation episodes. **Left**: the histogram over different options used by each agent; **Centre to right**: scatter plots of the action space, state space, and task information, colored by the corresponding option selected. Each of these spaces has been projected to $2D$ using t-SNE, except for the two-dimensional action space for Ball, which is plotted directly. For each case, care has been taken to choose a median / representative model out of 5 seeds.

A.4. Off-Policy Option Learning

In order to train in a more on-policy regime, we reduce the size of the replay buffer by two orders of magnitude and increase the ratio between data generation (actor steps) and data fitting (learner steps) by one order of magnitude. The resulting algorithm is run without any additional hyperparameter tuning to provide an insight into the effect of conditioning on action probabilities under options in the inference procedure. We can see that in the on-policy case the impact of this change is less pronounced. Across all cases, we were unable to generate significant performance gains by including action conditioning into the inference procedure.

	Scenario		Option entropy	Switch rate	Cluster score (actions)	Cluster score (states)	Cluster score (tasks)
Regular	Ball	No IA	2.105 ± 0.074	0.196 ± 0.010	-0.269 ± 0.058	-0.110 ± 0.025	-0.056 ± 0.011
		With IA	2.123 ± 0.066	0.346 ± 0.024	-0.056 ± 0.024	-0.164 ± 0.051	-0.057 ± 0.008
	Ant	No IA	1.583 ± 0.277	0.268 ± 0.043	-0.148 ± 0.034	-0.182 ± 0.068	-0.075 ± 0.011
		With IA	2.119 ± 0.073	0.303 ± 0.019	-0.053 ± 0.021	-0.066 ± 0.024	-0.052 ± 0.006
	Quad	No IA	1.792 ± 0.127	0.336 ± 0.019	-0.078 ± 0.064	-0.113 ± 0.035	-0.089 ± 0.050
		With IA	2.210 ± 0.037	0.403 ± 0.014	0.029 ± 0.029	-0.040 ± 0.003	-0.047 ± 0.006
Limited Switches	Ball	No IA	1.804 ± 0.214	0.020 ± 0.009	-0.304 ± 0.040	-0.250 ± 0.135	-0.131 ± 0.049
		With IA	2.233 ± 0.027	0.142 ± 0.015	-0.132 ± 0.035	-0.113 ± 0.043	-0.053 ± 0.003
	Ant	No IA	1.600 ± 0.076	0.073 ± 0.014	-0.124 ± 0.017	-0.155 ± 0.067	-0.084 ± 0.034
		With IA	2.222 ± 0.043	0.141 ± 0.015	-0.052 ± 0.011	-0.054 ± 0.014	-0.050 ± 0.007
	Quad	No IA	1.549 ± 0.293	0.185 ± 0.029	-0.075 ± 0.036	-0.126 ± 0.030	-0.112 ± 0.022
		With IA	2.231 ± 0.042	0.167 ± 0.025	-0.029 ± 0.029	-0.032 ± 0.004	-0.053 ± 0.009

Table 2: Quantitative results indicating the diversity of options used (entropy), and clustering accuracy in action and state spaces (silhouette score), with and without information asymmetry (IA), and with or without limited number of switches. Higher values indicate greater separability by option / component.

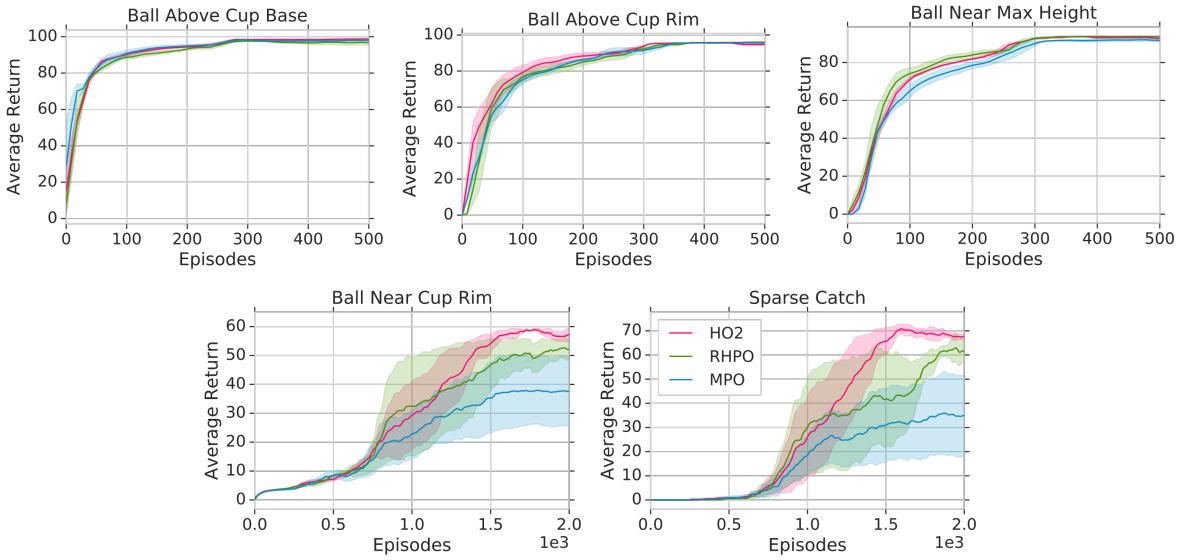

Figure 11: Complete results on pixel-based ball-in-cup experiments.

The complete results for all experiments with and without the action-conditional inference procedure can be found in Figure 14.

A.5. Trust-region Constraints

The complete results for all trust-region ablation experiments can be found in Figure 15.

With the exception of very high or very low constraints, the approach trains robustly, but performance drops considerably when we remove the constraint fully.

A.6. Single Time-Step vs Multi Time-Step Inference

To investigate the impact of probabilistic inference of posterior option distributions $\pi_H(o_t|h_t)$ along the whole sampled trajectory instead of using sampling-based approximations until the current timestep, we perform additional ablations displayed in Figure 16. Note that we are required to perform probabilistic inference for at least one step to use backpropagation through the inference step to update our policy components. Any completely sampling-based approach would require a different policy optimizer (e.g. via likelihood ratio or reparametrization trick) which would introduce additional compounding effects.

We compare HO2 with an ablated version where we do not compute the option probabilities along the trajectory following Equation 3 but instead use an approximation with only concrete option samples propagating across timesteps for all steps until the current step. To generate action samples, we therefore sample options for every timestep along

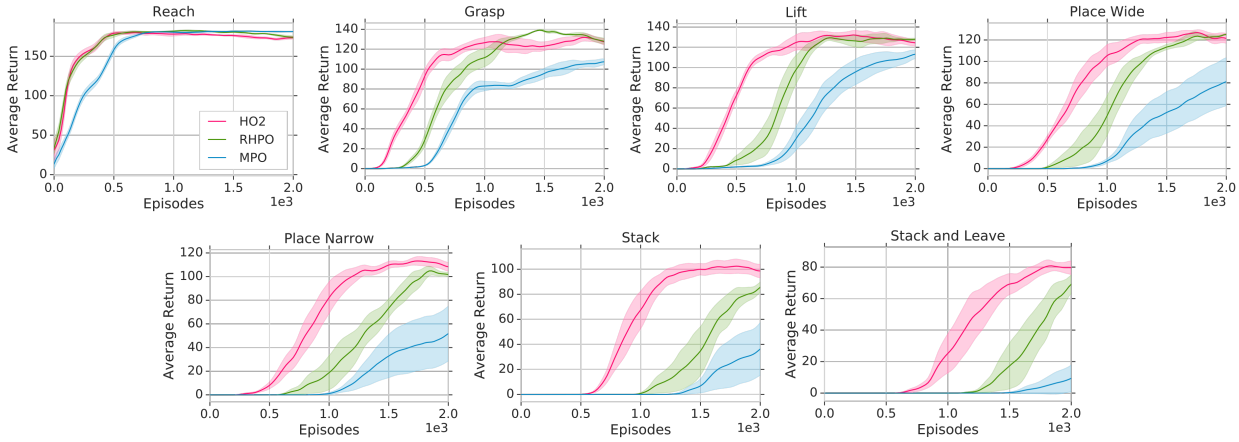

Figure 12: Complete results on pixel-based stacking experiments.

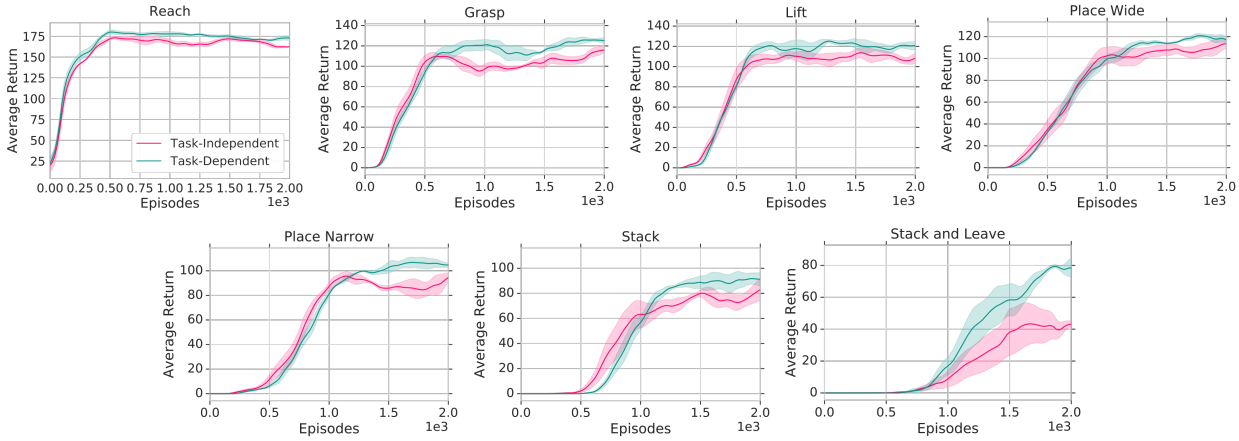

Figure 13: Complete results on multi-task block stacking with and without conditioning termination conditions on tasks.

a trajectory without keeping a complete distribution over options and sample actions only from the active option at every timestep. To determine the likelihood of actions and options for every timestep, we rely on Equation 2 based the sampled options of the previous timestep. By using samples and the critic-weighted update procedure from Equation 8, we can only generate gradients for the policy for the current timestep instead of backpropagating through the whole inference procedure. We find that using both samples from executed options reloaded from the buffer as well as new samples during learning can reduce performance depending on the domain. However, in the Hopper-v2 environment, sampling during learning performs slightly better than inferring options.

B. Additional Experiment Details

B.1. OpenAI Gym Experiments

All experiments are run with asynchronous learner and actors. We use a single actor and report performance over the number of transitions generated. Following (Wulfmeier et al., 2020), both HO2 and RHPO use different biases for the initial mean of all options or mixture components - distributed between minimum and maximum action output. This provides a small but non-negligible benefit and supports specialization of individual options. In line with our baselines (DAC (Zhang & Whiteson, 2019), IOPG (Smith et al., 2018), Option Critic (Bacon et al., 2017)) we use 4 options or mixture components for the OpenAI gym experiments. We run all experiments with 5 samples and report variance and mean. All experiments are run with a single actor in a distributed setting. The variant with limited switches limits to 2 switches over a sequence length of 8. Lower and higher values led to comparable results.

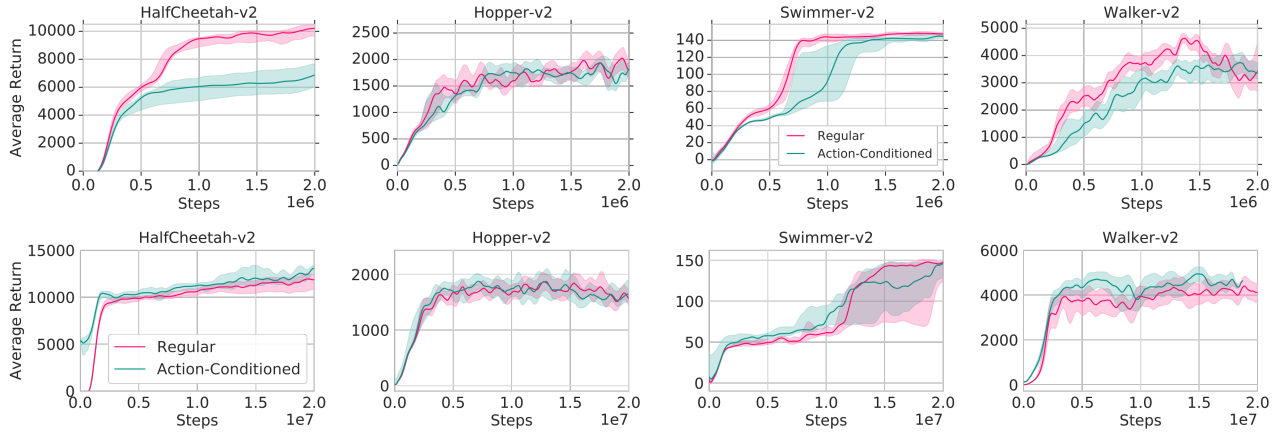

Figure 14: Complete results on OpenAI gym with and without conditioning component probabilities on past executed actions. For the off-policy (top) and on-policy case (bottom). The on-policy approaches uses data considerably less efficiently and the x-axis is correspondingly adapted.

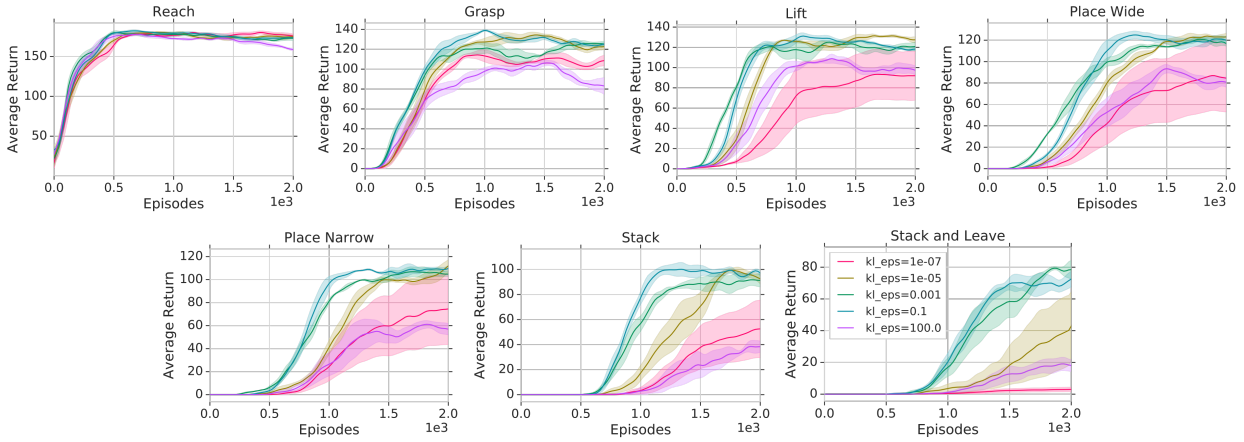

Figure 15: Complete results on block stacking with varying trust-region constraints for both termination conditions β and the high-level controller π_C.

B.2. Action and Temporal Abstraction Experiments

Shared across all algorithms, we use 3-layer convolutional policy and Q-function torsos with [128, 64, 64] feature channels, [(4, 4), (3, 3), (3, 3)] as kernels and stride 2. For all multitask domains, we build on information asymmetry and only provide task information as input to the high-level controller and termination conditions to create additional incentive for the options to specialize. The Q-function has access to all observations (see the corresponding tables in this section). We follow (Riedmiller et al., 2018; Wulfmeier et al., 2020) and assign rewards for all possible tasks to trajectories when adding data to the replay buffer independent of the generating policy.

Stacking The setup consists of a Sawyer robot arm mounted on a table and equipped with a Robotiq 2F-85 parallel gripper. In front of the robot there is a basket of size 20x20 cm which contains three cubes with an edge length of 5 cm (see Figure 4).

The agent is provided with proprioception information for the arm (joint positions, velocities and torques), and the tool center point position computed via forward kinematics. For the gripper, it receives the motor position and velocity, as well as a binary grasp flag. It also receives a wrist sensor's force and torque readings. Finally, it is provided with three RGB camera images at 64×64 resolution. At each timestep, a history of two previous observations (except for the images) is provided to the agent, along with the last two joint control commands. The observation space is detailed in Table 6. All stacking experiments are run with 50 actors in parallel and reported over the current episodes generated by any actor. Episode lengths are up to 600 steps.

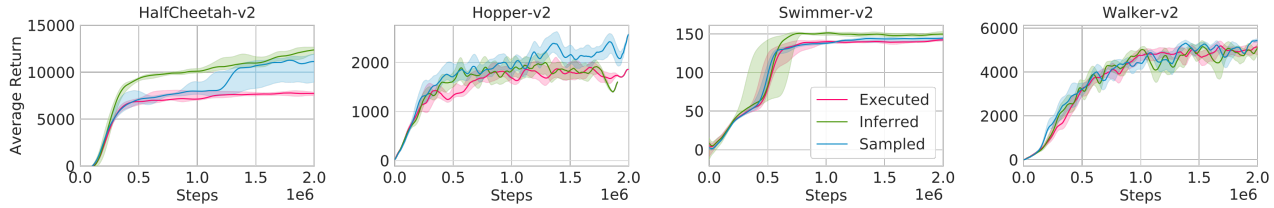

Figure 16: Ablation results comparing inferred options with sampled options during learning (sampled) and during execution (executed). The ablation is run with five actors instead of a single one as used in the OpenAI gym experiments in order to generate results faster.

Hyperparameters	HO2	RHPO	MPO
Policy net	256-256		
Number of actions samples	20		
Q function net	256-256		
Number of components	4		NA
ϵ	0.1		
ϵ_μ	5e-4		
ϵ_Σ	5e-5		
ϵ_α	1e-4		NA
ϵ_t	1e-4	NA	
Discount factor (γ)	0.99		
Adam learning rate	3e-4		
Replay buffer size	2e6		
Target network update period	200		
Batch size	256		
Activation function	elu		
Layer norm on first layer	Yes		
Tanh on output of layer norm	Yes		
Tanh on actions (Q-function)	Yes		
Sequence length	8		

Table 3: Hyperparameters - OpenAI gym

Hyperparameters	HO2	RHPO	MPO
Policy torso (shared across tasks)	256		512
Policy task-dependent heads	100 (cat.)		200
Policy shared heads	100 (comp.)		NA
Policy task-dependent terminations	100 (term.)	NA	NA
ϵ_μ	1e-3		
ϵ_Σ	1e-5		
ϵ_α	1e-4		NA
ϵ_t	1e-4	NA	
Number of action samples	20		
Q function torso (shared across tasks)	400		
Q function head (per task)	300		
Number of components	number of tasks		NA
Replay buffer size	1e6		
Target network update period	500		
Batch size	256		

Table 4: Hyperparameters. Values are taken from the OpenAI gym experiments with the above mentioned changes.

The robot arm is controlled in Cartesian velocity mode at 20Hz. The action space for the agent is 5-dimensional, as detailed in Table 5. The gripper movement is also restricted to a cubic volume above the basket using virtual walls.

$$stol(v, \epsilon, r) = \begin{cases} 1 & \text{iff } |v| < \epsilon \\ 1 - tanh^2\left(\frac{atanh(\sqrt{0.95})}{r}|v|\right) & \text{else} \end{cases} \quad (10)$$

$$slin(v, \epsilon_{min}, \epsilon_{max}) = \begin{cases} 0 & \text{iff } v < \epsilon_{min} \\ 1 & \text{iff } v > \epsilon_{max} \\ \frac{v - \epsilon_{min}}{\epsilon_{max} - \epsilon_{min}} & \text{else} \end{cases} \quad (11)$$

$$btol(v, \epsilon) = \begin{cases} 1 & \text{iff } |v| < \epsilon \\ 0 & \text{else} \end{cases} \quad (12)$$

Table 5: Action space for the Sawyer Stacking experiments.

Entry	Dims	Unit	Range
Translational Velocity in x, y, z	3	m/s	[-0.07, 0.07]
Wrist Rotation Velocity	1	rad/s	[-1, 1]
Finger speed	1	tics/s	[-255, 255]

- *REACH(G)*: $stol(d(TCP, G), 0.02, 0.15)$: Minimize the distance of the TCP to the green cube.
- *GRASP*: Activate grasp sensor of gripper ("inward grasp signal" of Robotiq gripper)

Table 6: Observations for the Sawyer Stacking experiments. The TCP's pose is represented as its world coordinate position and quaternion. In the table, m denotes meters, rad denotes radians, and q refers to a quaternion in arbitrary units (au).

Entry	Dims	Unit	History
Joint Position (Arm)	7	rad	2
Joint Velocity (Arm)	7	rad/s	2
Joint Torque (Arm)	7	Nm	2
Joint Position (Hand)	1	tics	2
Joint Velocity (Hand)	1	tics/s	2
Force-Torque (Wrist)	6	N, Nm	2
Binary Grasp Sensor	1	au	2
TCP Pose	7	m, au	2
Camera images	$3 \times 64 \times 64 \times 3$	R/G/B value	0
Last Control Command	8	rad/s, tics/s	2

- *LIFT(G)*: $slin(G, 0.03, 0.10)$
  Increase z coordinate of an object more than 3cm relative to the table.
- *PLACE_WIDE(G, Y)*: $stol(d(G, Y + [0, 0, 0.05]), 0.01, 0.20)$
  Bring green cube to a position 5cm above the yellow cube.
- *PLACE_NARROW(G, Y)*: $stol(d(G, Y + [0, 0, 0.05]), 0.00, 0.01)$:
  Like PLACE_WIDE(G, Y) but more precise.
- *STACK(G, Y)*: $btol(d_{xy}(G, Y), 0.03) * btol(d_z(G, Y) + 0.05, 0.01) * (1 - \textit{GRASP})$
  Sparse binary reward for bringing the green cube on top of the yellow one (with 3cm tolerance horizontally and 1cm vertically) and disengaging the grasp sensor.
- *STACK_AND_LEAVE(G, Y)*: $stol(d_z(TCP, G) + 0.10, 0.03, 0.10) * \textit{STACK(G, Y)}$
  Like STACK(G, Y), but needs to move the arm 10cm above the green cube.

Ball-In-Cup This task consists of a Sawyer robot arm mounted on a pedestal. A partially see-through cup structure with a radius of 11cm and height of 17cm is attached to the wrist flange. Between cup and wrist there is a ball bearing, to which a yellow ball of 4.9cm diameter is attached via a string of 46.5cm length (see Figure 4).

Most of the settings for the experiment align with the stacking task. The agent is provided with proprioception information for the arm (joint positions, velocities and torques), and the tool center point and cup positions computed via forward kinematics. It is also provided with two RGB camera images at 64×64 resolution. At each timestep, a history of two previous observations (except for the images) is provided to the agent, along with the last two joint control commands. The observation space is detailed in Table 8. All BIC experiments are run with 20 actors in parallel and reported over the current episodes generated by any actor. Episode lengths are up to 600 steps.

The position of the ball in the cup's coordinate frame is available for reward computation, but not exposed to the agent. The robot arm is controlled in joint velocity mode at 20Hz. The action space for the agent is 4-dimensional, with only 4 out of 7 joints being actuated, in order to avoid self-collision. Details are provided in Table 5.

Table 7: Action space for the Sawyer Ball-in-Cup experiments.

Entry	Dims	Unit	Range
Rotational Joint Velocity for joints 1, 2, 6 and 7	4	rad/s	[-2, 2]

Table 8: Observations for the Sawyer Ball-in-Cup experiments. In the table, m denotes meters, rad denotes radians, and q refers to a quaternion in arbitrary units (au). Note: the joint velocity and command represent the robot's internal state; the 3 degrees of freedom that were fixed provide a constant input of 0.

Entry	Dims	Unit
Joint Position (Arm)	7	rad
Joint Velocity (Arm)	7	rad/s
TCP Pose	7	m, au
Camera images	$2 \times 64 \times 64 \times 3$	R/G/B value
Last Control Command	7	rad/s

Let B_A be the Cartesian position in meters of the ball in the cup's coordinate frame (with an origin at the center of the cup's bottom), along axes $A \in \{x, y, z\}$.

- *CATCH*: $0.17 > B_z > 0$ and $||B_{xy}||_2 < 0.11$
  Binary reward if the ball is inside the volume of the cup.
- *BALL_ABOVE_BASE*: $B_z > 0$
  Binary reward if the ball is above the bottom plane of the cup.
- *BALL_ABOVE_RIM*: $B_z > 0.17$
  Binary reward if the ball is above the top plane of the cup.
- *BALL_NEAR_MAX*: $B_z > 0.3$
  Binary reward if the ball is near the maximum possible height above the cup.

- *BALL_NEAR_RIM*: $1 - tanh^2(\frac{atanh(\sqrt{0.95})}{0.5} \times ||B_{xyz} - (0, 0, 0.17)||_2)$
  Shaped distance of the ball to the center of the cup opening (0.95 loss at a distance of 0.5).

B.3. Pre-training and Sequential Transfer Experiments

The sequential transfer experiments are performed with the same settings as their multitask equivalents. However, they rely on a pre-training step in which we take all but the final task in each domain and train HO2 to pre-train options which we then transfer with a new high-level controller on the final task. Fine-tuning of the options is enabled as we find that it produces slightly better performance. Only data used for the final training step is reported but all both approaches were trained for the same amount of data during pretraining until convergence. The variant with limited switches limits to 4 switches over a sequence length of 16.

B.4. Locomotion experiments

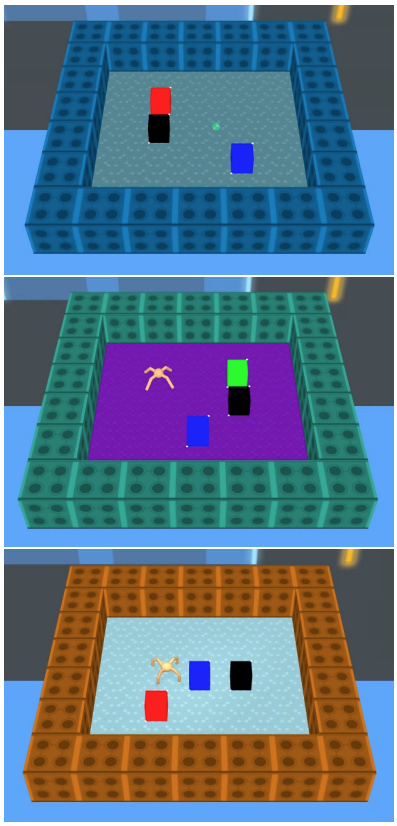

Figure 17: The environment used for simple locomotion tasks with Ball (top), Ant (center) and Quadruped (bottom).

Figure 17 shows examples of the environment for the different bodies used. In addition to proprioceptive agent state information (which includes the body height, position of the end-effectors, the positions and velocities of its joints and sensor readings from an accelerometer, gyroscope and velocimeter attached to its torso), the state space also includes the ego-centric coordinates of all target locations and a categorical index specifying the task of interest. Table 9 contains an overview of the observations and action dimensions for this task. The agent receives a sparse reward of $+60$ if part of its body reaches a square surrounding the predicate location, and 0 otherwise. Both the agent spawn location and target locations are randomized at the start of each episode, ensuring that the agent must use both the task index and target locations to solve the task.

Table 9: Observations for the *go to one of 3 targets* task with Ball, Ant, and Quadruped.

Entry	Dimensionality
Task Index	3
Target locations	9
Proprioception (Ball)	16
Proprioception (Ant)	41
Proprioception (Quad)	57
Action Dim (Ball)	2
Action Dim (Ant)	8
Action Dim (Quad)	12

C. Additional Derivations

In this section we explain the derivations for training option policies with options parameterized as Gaussian distributions. Each policy improvement step is split into two parts: non-parametric and parametric update.

C.1. Non-parametric Option Policy Update

In order to obtain the non-parametric policy improvement we optimize the following equation:

$$\max_q \mathbb{E}_{h_t \sim p(h_t)} \left[\mathbb{E}_{a_t, o_t \sim q} \left[Q_\phi(s_t, a_t, o_t) \right] \right]$$
$$s.t. \mathbb{E}_{h_t \sim p(h_t)} \left[\mathrm{KL}(q(\cdot|h_t), \pi_\theta(\cdot|h_t)) \right] < \epsilon_E$$
$$s.t. \mathbb{E}_{h_t \sim p(h_t)} \left[\mathbb{E}_{q(a_t, o_t|h_t)} \left[1 \right] \right] = 1.$$

for each step t of a trajectory, where $h_t = \{s_t, a_{t-1}, s_{t-1}, ...a_0, s_0\}$ represents the history of states and actions and $p(h_t)$ describes the distribution over histories for timestep t, which in practice are approximated via the use of a replay buffer $\mathcal{D}$. When sampling h_t, the state s_t is the first element of the history. The inequality constraint describes the maximum allowed KL divergence between intermediate update and previous parametric policy, while the equality constraint simply ensures that the intermediate update represents a normalized distribution.

Subsequently, in order to render the following derivations

more intuitive, we replace the expectations and explicitly use integrals. The Lagrangian $L(q, \eta, \gamma)$ can now be formulated as

$$\begin{aligned} L(q,\eta,\gamma) = &\iiint p(h_t) q(a_t, o_t|h_t) Q_\phi(s_t, a_t, o_t) \quad (13) \\ &\qquad \mathrm{d}o_t \, \mathrm{d}a_t \, \mathrm{d}h_t \\ &+\eta \Big(\epsilon_E - \iiint p(h_t) q(a_t, o_t|h_t) \log \frac{q(a_t, o_t|h_t)}{\pi_\theta(a_t, o_t|h_t)} \\ &\qquad \mathrm{d}o_t \, \mathrm{d}a_t \, \mathrm{d}h_t \Big) \\ &+\gamma \left(1 - \iiint p(h_t) q(a_t, o_t|h_t) \, \mathrm{d}o_t \, \mathrm{d}a_t \, \mathrm{d}h_t \right). \end{aligned}$$

Next to maximize the Lagrangian with respect to the primal variable q, we determine its derivative as,

$$\begin{aligned} \frac{\partial L(q,\eta,\gamma)}{\partial q} = &Q_\phi(a_t, o_t, s_t) - \eta \log q(a_t, o_t|h_t) \\ &+\eta \log \pi_\theta(a_t, o_t|h_t) - \eta - \gamma. \end{aligned}$$

In the next step, we can set the left hand side to zero and rearrange terms to obtain

$$\begin{aligned} q(a_t, o_t|h_t) = \pi_\theta(a_t, o_t|h_t) \exp\left(\frac{Q_\phi(s_t, a_t, o_t)}{\eta} \right) \\ \exp\left(-\frac{\eta + \gamma}{\eta} \right). \end{aligned}$$

The last exponential term represents a normalization constant for q, which we can formulate as

$$\begin{aligned} \frac{\eta + \gamma}{\eta} = \log \Bigg(&\iint \pi_\theta(a_t, o_t|h_t) \\ &\exp\left(\frac{Q_\phi(s_t, a_t, o_t)}{\eta} \right) \mathrm{d}o_t \, \mathrm{d}a_t \Bigg). \end{aligned} \quad (14)$$

In order to obtain the dual function $g(\eta)$, we insert the solution for the primal variable into the Lagrangian in Equation 13 which yields

$$\begin{aligned} L(q,\eta,\gamma) = &\iiint p(h_t) q(a_t, o_t|h_t) Q_\phi(s_t, a_t, o_t) \\ &\qquad \mathrm{d}o_t \, \mathrm{d}a_t \, \mathrm{d}h_t \\ &+\eta \Big(\epsilon_E - \iiint p(h_t) q(a_t, o_t|h_t) \\ &\log \frac{\pi_\theta(a_t, o_t|h_t) \exp\left(\frac{Q_\phi(s_t, a_t, o_t)}{\eta} \right) \exp\left(-\frac{\eta+\gamma}{\eta} \right)}{\pi_\theta(a_t, o_t|h_t)} \mathrm{d}o_t \, \mathrm{d}a_t \, \mathrm{d}h_t \Big) \\ &+\gamma \left(1 - \iiint p(h_t) q(a_t, o_t|h_t) \, \mathrm{d}o_t \, \mathrm{d}a_t \, \mathrm{d}h_t \right). \end{aligned}$$

We expand the equation and rearrange to obtain

$$\begin{aligned} L(q,\eta,\gamma) = &\iiint p(h_t) q(a_t, o_t|h_t) Q_\phi(s_t, a_t, o_t) \\ &\qquad \mathrm{d}o_t \, \mathrm{d}a_t \, \mathrm{d}h_t \\ &- \eta \iiint p(h_t) q(a_t, o_t|h_t) \Big[\frac{Q_\phi(s_t, a_t, o_t)}{\eta} \\ &+ \log \pi_\theta(a_t, o_t|h_t) - \frac{\eta + \gamma}{\eta} \Big] \mathrm{d}o_t \, \mathrm{d}a_t \, \mathrm{d}h_t \\ &+ \eta \epsilon_E + \eta \iiint p(h_t) q(a_t, o_t|h_t) \\ &\qquad \log \pi_\theta(a_t, o_t|h_t) \, \mathrm{d}o_t \, \mathrm{d}a_t \, \mathrm{d}h_t \\ &+ \gamma \left(1 - \iiint p(h_t) q(a_t, o_t|h_t) \, \mathrm{d}o_t \, \mathrm{d}a_t \, \mathrm{d}h_t \right). \end{aligned}$$

In the next step, most of the terms cancel out and after additional rearranging of the terms we obtain

$$L(q,\eta,\gamma) = \eta \epsilon_E + \eta \int p(h_t) \frac{\eta + \gamma}{\eta} \, \mathrm{d}h_t.$$

We have already calculated the term inside the integral in Equation 14, which we now insert to obtain

$$\begin{aligned} g(\eta) = &\min_q L(q,\eta,\gamma) \quad (15) \\ = &\eta \epsilon_E + \eta \int p(h_t) \log \Bigg(\iint \pi_\theta(a_t, o_t|h_t) \\ &\qquad \exp\left(\frac{Q_\phi(s_t, a_t, o_t)}{\eta} \right) \mathrm{d}o_t \, \mathrm{d}a_t \Bigg) \mathrm{d}h_t \\ = &\eta \epsilon_E + \eta \mathbb{E}_{h_t \sim p(h_t)} \Big[\log \Big(\mathbb{E}_{a_t, o_t \sim \pi_\theta} \Big[\\ &\qquad \exp\left(\frac{Q_\phi(s_t, a_t, o_t)}{\eta} \right) \Big] \Big) \Big]. \end{aligned}$$

The dual in Equation 15 can finally be minimized with respect to η based on samples from the replay buffer and policy.

C.2. Parametric Option Policy Update

After obtaining the non-parametric policy improvement, we can align the parametric option policy to the current non-parametric policy. As the non-parametric policy is represented by a set of samples from the parametric policy with additional weighting, this step effectively employs a type of critic-weighted maximum likelihood estimation. In addition, we introduce regularization based on a distance function $\mathcal{T}$ which has a trust-region effect for the update and stabilizes learning.

$$\begin{aligned}\theta_{new} &= \arg\min_{\theta} \mathbb{E}_{h_t\sim p(h_t)}\Big[\mathrm{KL}\big(q(a_t,o_t|h_t)\|\pi_\theta(a_t,o_t|h_t)\big)\Big] \\ &= \arg\min_{\theta} \mathbb{E}_{h_t\sim p(h_t)}\Big[\mathbb{E}_{a_t,o_t\sim q}\Big[\log q(a_t,o_t|h_t) \\ &\qquad - \log\pi_\theta(a_t,o_t|h_t)\Big]\Big] \\ &= \arg\max_{\theta} \mathbb{E}_{h_t\sim p(h_t),a_t,o_t\sim q}\Big[\log\pi_\theta(a_t,o_t|h_t)\Big], \\ &\quad \text{s.t. } \mathbb{E}_{h_t\sim p(h_t)}\Big[\mathcal{T}(\pi_{\theta_{new}}(\cdot|h_t)|\pi_\theta(\cdot|h_t))\Big] < \epsilon_M,\end{aligned}$$

where $h_t \sim p(h_t)$ is a trajectory segment, which in practice sampled from the dataset $\mathcal{D}$, $\mathcal{T}$ is an arbitrary distance function between the new policy and the previous policy. ϵ_M denotes the allowed change for the policy. We again employ Lagrangian Relaxation to enable gradient based optimization of the objective, yielding the following primal:

$$\begin{aligned}\max_{\theta}\min_{\alpha>0} L(\theta,\alpha) &= \mathbb{E}_{h_t\sim p(h_t),a_t,o_t\sim q}\Big[\log\pi_\theta(a_t,o_t|h_t)\Big] \\ &+\alpha\Big(\epsilon_M - \mathbb{E}_{h_t\sim p(h_t)}\big[\mathcal{T}(\pi_{\theta_{new}}(\cdot|h_t),\pi_\theta(\cdot|h_t))\big]\Big).\end{aligned} \tag{16}$$

We can solve for θ by iterating the inner and outer optimization programs independently. In practice we find that it is most efficient to update both in parallel.

We also define the following distance function between old and new option policies

$$\mathcal{T}(\pi_{\theta_{new}}(\cdot|h_t),\pi_\theta(\cdot|h_t)) = \mathcal{T}_H(h_t) + \mathcal{T}_T(h_t) + \mathcal{T}_L(h_t)$$

$$\begin{aligned}\mathcal{T}_H(h_t) &= \mathrm{KL}(\mathrm{Cat}(\{\alpha^j_{\theta_{new}}(h_t)\}_{j=1\dots M})\| \\ &\qquad \mathrm{Cat}(\{\alpha^j_{\theta}(h_t)\}_{j=1\dots M})) \\ \mathcal{T}_T(h_t) &= \frac{1}{M}\sum_{j=1}^{M}\mathrm{KL}(\mathrm{Cat}(\{\beta^{ij}_{\theta_{new}}(h_t)\}_{j=1\dots 2})\| \\ &\qquad \mathrm{Cat}(\{\beta^{ij}_{\theta}(h_t)\}_{j=1\dots 2})) \\ \mathcal{T}_L(h_t) &= \frac{1}{M}\sum_{j=1}^{M}\mathrm{KL}(\mathcal{N}(\mu^j_{\theta_{new}}(h_t),\Sigma^j_{\theta_{new}}(h_t))\| \\ &\qquad \mathcal{N}(\mu^j_{\theta}(h_t),\Sigma^j_{\theta}(h_t)))\end{aligned}$$

where $\mathcal{T}_H$ evaluates the KL between the categorical distributions of the high-level controller, $\mathcal{T}_T$ is the average KL between the categorical distributions of the all termination conditions, and $\mathcal{T}_L$ corresponds to the average KL across Gaussian components. In practice, we can exert additional control over the convergence of model components by applying different ϵ_M to different model parts (high-level controller, termination conditions, options).

C.3. Transition Probabilities for Option and Switch Indices

The transitions for option o and switch index n are given by:

$$p(o_t,n_t|s_t,o_{t-1},n_{t-1}) = \begin{cases}(1-\beta(s_t,o_{t-1})) & \text{if } n_t = n_{t-1}, o_t = o_{t-1} \\ \beta(s_t,o_{t-1})\pi^C(o_t|s_t) & \text{if } n_t = n_{t-1}+1 \\ 0 & \text{otherwise}\end{cases} \tag{17}$$