Data-efficient Hindsight Off-policy Option Learning

Markus Wulfmeier [1] **Dushyant Rao** [1] **Roland Hafner** [1] **Thomas Lampe** [1] **Abbas Abdolmaleki** [1] **Tim Hertweck** [1] **Michael Neunert** [1] **Dhruva Tirumala** [1] **Noah Siegel** [1] **Nicolas Heess** [1] **Martin Riedmiller** [1]

Abstract

We introduce Hindsight Off-policy Options (HO2), a data-efficient option learning algorithm. Given any trajectory, HO2 infers likely option choices and backpropagates through the dynamic programming inference procedure to robustly train all policy components off-policy and end-to-end. The approach outperforms existing option learning methods on common benchmarks. To better understand the option framework and disentangle benefits from both temporal and action abstraction, we evaluate ablations with flat policies and mixture policies with comparable optimization. The results highlight the importance of both types of abstraction as well as off-policy training and trust-region constraints, particularly in challenging, simulated 3D robot manipulation tasks from raw pixel inputs. Finally, we intuitively adapt the inference step to investigate the effect of increased temporal abstraction on training with pre-trained options and from scratch.

1. Introduction

Deep reinforcement learning has seen numerous successes in recent years (Silver et al., 2017; OpenAI et al., 2018; Vinyals et al., 2019), but still faces challenges in domains where data is limited or expensive. One candidate solution to address the challenges and improve data efficiency is to impose hierarchical policy structures. By dividing an agent into a combination of low-level and high-level controllers, the options framework (Sutton et al., 1999; Precup, 2000) introduces a form of action abstraction, effectively reducing the high-level controller's task to choosing from a discrete set of reusable sub-policies. The framework further enables temporal abstraction by explicitly modelling the temporal continuation of low-level behaviors. Unfortunately, in practice, hierarchical control schemes often introduce technical challenges, including a tendency to learn degenerate solutions preventing the agent from using its full capacity (Harb et al., 2018), undesirable trade-offs between learning efficiency and final performance (Harutyunyan et al., 2019), or the increased variance of updates (Precup, 2000). Additional challenges in off-policy learning for hierarchical approaches (Precup et al., 2006) led to a focus of recent works on the on-policy setting, forgoing the considerable improvements in data efficiency often connected to off-policy methods.

We propose an approach to address these drawbacks, Hindsight Off-policy Options (HO2): a method for data-efficient, robust, off-policy option learning. The algorithm simultaneously learns a high-level controller and low-level options via a single end-to-end optimization procedure. It improves data efficiency by leveraging off-policy learning and inferring distributions over option for trajectories in hindsight to maximize the likelihood of good actions and options.

To facilitate off-policy learning the algorithm does not condition on executed options but treats these as latent variables during optimization and marginalizes over all options to compute the exact likelihood. HO2 backpropagates through the resulting dynamic programming inference graph (conceptually related to (Rabiner, 1989; Shiarlis et al., 2018; Smith et al., 2018)) to enable the training of all policy components from trajectories, independent of the executed option. As an additional benefit, the formulation of the inference graph allows to impose intuitive, hard constraints on the option termination frequency, thereby regularizing the learned solution (and encouraging temporally-extended behaviors) independently of the scale of the reward.

The policy update follows an expectation-maximization perspective and generates an intermediate, non-parametric policy, which is adapted to maximize agent performance. This enables the update of the parametric policy to rely on simple weighted maximum likelihood, without requiring further approximations such as Monte Carlo estimation or continuous relaxation (Li et al., 2019). Finally, the updates are stabilized using adaptive trust-region constraints, demonstrating the importance of robust policy optimization for hierarchical reinforcement learning (HRL) in line with recent work on on-policy option learning (Zhang & Whiteson, 2019).

[1]DeepMind, London, United Kingdom. Correspondence to: Markus Wulfmeier <mwulfmeier@google.com>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

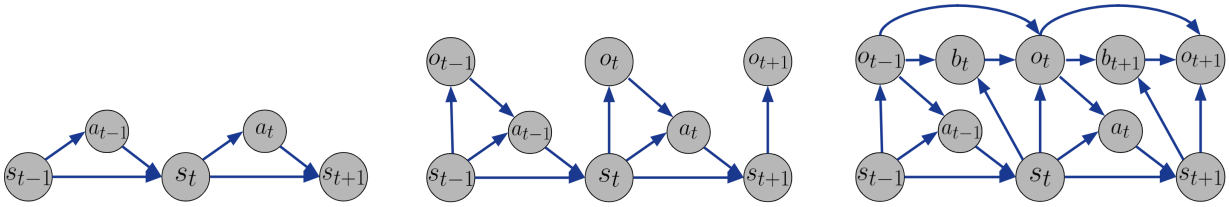

Figure 1: Graphical model for flat policies (left), mixture policies (middle) - introducing a type of action abstraction, and option policies (right) - adding temporal abstraction via autoregressive options. While the action a is solely dependent on the state s for flat policies, mixture policies introduce the additional component or option o which affects the actions (following Equation 1). Option policies do not change the direct dependencies for actions but instead affect the options themselves, which are now also dependent on the previous option and its potential termination b (following Equation 2).

We experimentally compare HO2 to prior option learning methods. By treating options as latent variables in off-policy learning and enabling backpropagation through the inference procedure, HO2 demonstrates to be more efficient than prior approaches such as the Option-Critic (Bacon et al., 2017) or DAC (Zhang & Whiteson, 2019). HO2additionally outperforms IOPG (Smith et al., 2018), which considers a similar perspective but still builds on on-policy training. To better understand different abstractions in option learning, we compare with corresponding policy optimization methods for flat policies (Abdolmaleki et al., 2018a) and mixture policies without temporal abstraction (Wulfmeier et al., 2020) thereby allowing us to isolate the benefits of both action and temporal abstraction. Both properties demonstrate particular relevance in more demanding simulated robot manipulation tasks from raw pixel inputs. We further perform extensive ablations to evaluate the impact of trust-region constraints, off-policyness, option decomposition, and the benefits of maximizing temporal abstraction when using pre-trained options versus learning from scratch.

Our main contributions include:

- A robust, efficient off-policy option learning algorithm enabled by a probabilistic inference perspective on HRL. The method outperforms existing option learning methods on common benchmarks and demonstrates benefits on pixel-based 3D robot manipulation tasks.
- An intuitive technique to further encourage temporal abstraction beyond the core method, using the inference graph to constrain option switches without additional weighted loss terms.
- A careful analysis to improve our understanding of the options framework by isolating the impact of action abstraction and temporal abstraction.
- Further ablation and analysis of several algorithmic choices: trust-region constraints, off-policy versus on-policy data, option decomposition, and the importance of temporal abstraction with pre-trained options versus learning from scratch.

2. Method

We start by considering a reinforcement learning setting with an agent operating in a Markov Decision Process (MDP) consisting of the state space $\mathcal{S}$, the action space $\mathcal{A}$, and the transition probability $p(s_{t+1}|s_t, a_t)$ of reaching state s_{t+1} from state s_t when executing action a_t. The agent's behavior is commonly described as a conditional distribution with actions a_t drawn from the agent's policy $\pi(a_t|s_t)$. Jointly, the transition dynamics and policy induce the marginal state visitation distribution $p(s_t)$. The discount factor γ together with the reward $r_t = r\left(s_t, a_t\right)$ gives rise to the expected return, which the agent aims to maximize: $J(\pi) = \mathbb{E}_{p(s_t),\pi(a_t|s_t)}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$.

2.1. Policy Types

Option policies introduce temporal and action abstraction in comparison to commonly-used flat Gaussian policies. Our goal in this work is not only to introduce this additional structure to improve data efficiency but to further understand the impact of the different abstractions. For this purpose, we further study mixture distributions. They represent an intermediate case with only action abstraction, as described in Figure 1.

We begin by covering both policy types in the following paragraphs. First, we focus on computing likelihoods of actions (and options) under a policy. Then, we describe the proposed critic-weighted maximum likelihood algorithm to train hierarchical policies.

Mixture Policies This type extends flat policies $\pi(a_t|s_t)$ by introducing a high-level controller that samples from multiple options (low-level policies) independently at each timestep (Figure 1). The joint probability of actions and options is given as:

$$\pi_\theta(a_t, o_t|s_t) = \pi^L\left(a_t|s_t, o_t\right)\pi^H\left(o_t|s_t\right), \qquad (1)$$

where π^H and π^L respectively represent high-level policy (which for the mixture is equal to a Categorical distribution

$\pi^H(o_t|s_t) = \pi^C(o_t|s_t)$) and low-level policy (components of the resulting mixture distribution), and o is the index of the sub-policy or mixture component.

Option Policies This type extends mixture policies by incorporating temporal abstraction. We follow the semi-MDP and *call-and-return* option model (Sutton et al., 1999), defining an option as a triple $(I(s_t, o_t), \pi^L(a_t|s_t, o_t), \beta(s_t, o_t))$. The initiation condition I describes an option's probability to start in a state and is simplified to $I(s_t, o_t) = 1 \forall s_t \in \mathcal{S}$ following (Bacon et al., 2017; Zhang & Whiteson, 2019). The termination condition $b_t \sim \beta(s_t, o_t)$ denotes a Bernoulli distribution describing the option's probability to terminate in any given state, and the action distribution for a given option is modelled by $\pi^L(a_t|s_t, o_t)$. Every time the agent observes a state, the current option's termination condition is sampled. If subsequently no option is active, a new option is sampled from the controller $\pi^C(o_t|s_t)$. Finally, we sample from either the continued or new option to generate a new action. The resulting transition probabilities between options are described by

$$p(o_t|s_t, o_{t-1}) = \begin{cases} 1 - \beta(s_t, o_{t-1})(1 - \pi^C(o_t|s_t)) & \text{if } o_t = o_{t-1} \\ \beta(s_t, o_{t-1})\pi^C(o_t|s_t) & \text{otherwise} \end{cases} \tag{2}$$

During interaction in an environment, an agent samples individual options. However, during learning HO2 takes a probabilistic inference perspective with options as latent variables and states and actions as observed variables. This allows us to infer likely options over a whole trajectory in hindsight, leading to efficient intra-option learning (Precup, 2000) for all options independently of the executed option. This is particularly relevant for off-policy learning, as options can change between data generation and learning.

Following the graphical model in Figure 1 and corresponding transition probabilities in Equation 2, the probability of being in option o_t at timestep t across a trajectory $h_t = \{s_t, a_{t-1}, s_{t-1}, ...s_0, a_0\}$ is determined in a recursive manner based on the previous timestep's option probabilities. For the first timestep, the probabilities are given by the high-level controller $\pi^H(o_0|h_0) = \pi^C(o_0|s_0)$. For all consecutive steps are computed as follows for M options:

$$\tilde{\pi}^H(o_t|h_t) = \sum_{o_{t-1}=1}^{M} \left[p(o_t|s_t, o_{t-1}) \pi^H(o_{t-1}|h_{t-1}) \pi^L(a_{t-1}|s_{t-1}, o_{t-1}) \right] \tag{3}$$

The distribution is normalized at each timestep following $\pi^H(o_t|h_t) = \tilde{\pi}^H(o_t|h_t) / \sum_{o'_t=1}^{M} \tilde{\pi}^H(o'_t|h_t)$. Performing this exact marginalization at each timestep is much more efficient than computing independently over all possible sequences of options and reduces variance compared to sampling-based approximations.

Building on the option probabilities, Equation 4 conceptualizes the connection between mixture and option policies.

$$\pi_\theta(a_t, o_t|h_t) = \pi^L(a_t|s_t, o_t) \pi^H(o_t|h_t) \tag{4}$$

In both cases, the low-level policies π^L only depend on the current state. However, where mixtures only depend on the current state s_t for the high-level probabilities π^H, with options we can take into account compressed information about the history h_t as facilitated by the previous timestep's distribution over options $\pi^H(o_{t-1}|h_{t-1})$.

This dynamic programming formulation in Equation 3 enables the exact computation of the likelihood of actions and options along off-policy trajectories. We can use automatic differentiation in modern deep learning frameworks (e.g. (Abadi et al., 2016)) to backpropagate through the graph and determine the gradient updates for all policy parameters.

2.2. Agent Updates

We continue by describing the policy improvement algorithm, which uses the previously determined option probabilities. The three main steps are: 1) update the critic (Eq. 5); 2) generate an intermediate, non-parametric policy based on the updated critic (Eq. 6); 3) update the parametric policy to align to the non-parametric improvement (Eq. 8). By handling the maximization of expected returns with a closed-loop solution for a non-parametric intermediate policy, the update of the parametric policy can build on simple, weighted maximum likelihood. In essence, we do not rely on differentiating an expectation over a distribution with respect to parameters of the distribution. This enables training a broad range of distributions (including discrete ones) without further approximations such as required when the update relies on the reparametrization trick (Li et al., 2019).

Policy Evaluation In comparison to prior work on training mixture policies (Wulfmeier et al., 2020), the critic for option policies is a function of s, a, and o since the current option influences the likelihood of future actions and thus rewards. Note that even though we express the policy as a function of the history h_t, Q is a function of o_t, s_t, a_t, since these are sufficient to render the future trajectory independent of the past (see the graphical model in Figure 1). We define the TD(0) objective as

$$\min_\phi L(\phi) = \mathbb{E}_{s_t, a_t, o_t \sim \mathcal{D}} \left[\left(Q_\mathrm{T} - Q_\phi(s_t, a_t, o_t) \right)^2 \right], \tag{5}$$

where the current states, actions, and options are sampled from the current replay buffer $\mathcal{D}$. For the 1-step target $Q_\mathrm{T} = r_t + \gamma \mathbb{E}_{s_{t+1}, a_{t+1}, o_{t+1}}[Q'(s_{t+1}, a_{t+1}, o_{t+1})]$, the expectation over the next state is approximated with the sample s_{t+1} from the replay buffer, and we estimate

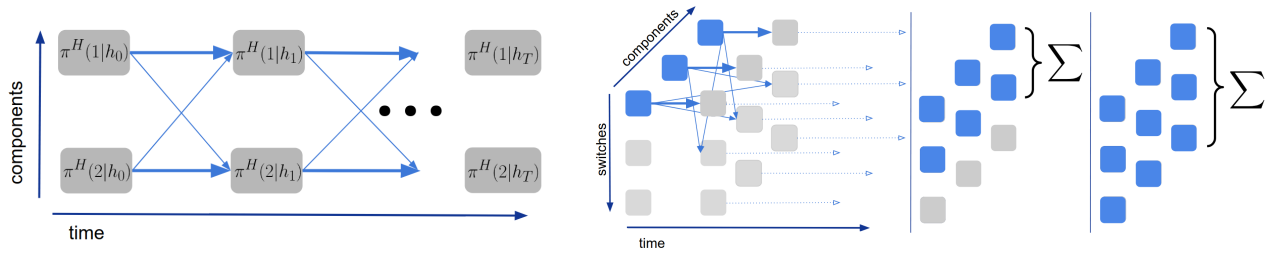

Figure 2: Representation of the dynamic programming forward pass - bold arrows represent connections without switching. Left: example with two options. Right: extension of the graph to explicitly count the number of switches. Marginalization over the dimension of switches determines component probabilities. By limiting over which nodes to sum at every timestep, the optimization can be targeted to fewer switches and more consistent option execution.

the value by sampling actions and options according to $a_{t+1}, o_{t+1} \sim \pi'(\cdot|h_{t+1})$ following Equation 4. π' and Q' respectively represent target networks for policy and critic which are used to stabilize training.

Policy Improvement We follow an Expectation-Maximization procedure similar to (Wulfmeier et al., 2020; Abdolmaleki et al., 2018b), which first computes an improved non-parametric policy and then updates the parametric policy to match this target. In comparison to prior work, the policy does not only depend on the current state s_t but also on compressed information about the rest of the previous trajectory h_t, building on Equation 3. Given the critic, all we require to optimize option policies is the ability to sample from the policy and determine the log-likelihood (gradient) of actions and options under the policy. The first step provides us with a non-parametric policy $q(a_t, o_t|h_t)$.

$$\begin{aligned} \max_q J(q) = \; & \mathbb{E}_{a_t,o_t\sim q, h_t\sim\mathcal{D}}\big[Q_\phi(s_t, a_t, o_t)\big], \\ \text{s.t. } & \mathbb{E}_{h_t\sim\mathcal{D}}\Big[\mathrm{KL}\big(q(\cdot|h_t)\|\pi_\theta(\cdot|h_t)\big)\Big] \leq \epsilon_E, \end{aligned} \tag{6}$$

where $\mathrm{KL}(\cdot\|\cdot)$ denotes the Kullback-Leibler divergence, and ϵ_E defines a bound on the KL. We can find the solution to the constrained optimization problem in Equation 6 in closed-form and obtain

$$q(a_t, o_t|h_t) \propto \pi_\theta(a_t, o_t|h_t) \exp\left(Q_\phi(s_t, a_t, o_t)/\eta\right). \tag{7}$$

Practically speaking, this step computes samples from the previous policy and weights these based on the corresponding temperature-calibrated values of the critic. The temperature parameter η is computed following the dual of the Lagrangian. The derivation and final form of the dual can be found in Appendix C.1, Equation 15.

To align the parametric to the improved non-parametric policy in the second step, we minimize their KL divergence.

$$\begin{aligned} \theta = \arg\min_\theta \; & \mathbb{E}_{h_t\sim\mathcal{D}}\Big[\mathrm{KL}\big(q(\cdot|h_t)\|\pi_\theta(\cdot|h_t)\big)\Big], \\ \text{s.t. } & \mathbb{E}_{h_t\sim\mathcal{D}}\Big[\mathcal{T}\big(\pi_{\theta_+}(\cdot|h_t)\|\pi_\theta(\cdot|h_t)\big)\Big] \leq \epsilon_M \end{aligned} \tag{8}$$

The distance function $\mathcal{T}$ in Equation 8 has a trust-region effect and stabilizes learning by constraining the change in the parametric policy. The computed option probabilities from Equation 3 are used in Equation 7 to enable sampling of options as well as Equation 8 to determine and maximize the likelihood of samples under the policy. We can apply Lagrangian relaxation again and solve the primal as detailed in Appendix C.2. Finally, we describe the complete pseudo-code for HO2 in Algorithm 1.

Note that both Gaussian and mixture policies have been trained in prior work via methods relying on critic-weighted maximum likelihood (Abdolmaleki et al., 2018a; Wulfmeier et al., 2020). By comparing with the extension towards option policies described above, we will make use of this connection to isolate the impact of action abstraction and temporal abstraction in the option framework in Section 3.2.

2.3. Maximizing Temporal Abstraction

Persisting with each option over longer time periods can help to reduce the search space and simplify exploration (Sutton et al., 1999; Harb et al., 2018). Previous approaches (e.g. (Harb et al., 2018)) rely on additional weighted loss terms which penalize option transitions.

In addition to the main HO2 algorithm, we introduce an extension mechanism to explicitly limit the maximum number of switches between options along a trajectory to increase temporal abstraction. In comparison to additional loss terms, a parameter for the maximum number of switches can be chosen independently of the reward scale of an environment and provides an intuitive semantic interpretation, both aspects simplify manual adaptation.

We extend the 2D graph for computing option probabilities

Algorithm 1 Hindsight Off-policy Options

input: initial parameters for θ, η and ϕ, KL regularization parameters ϵ, set of trajectories τ
repeat
  sample trajectories τ from replay buffer
  // forward pass along sampled trajectories
  determine component probabilities $\pi^H(o_t|h_t)$ (Eq. 3)
  sample actions a_j and options o_j from $\pi_\theta(\cdot|h_t)$ (Eq. 4) to estimate expectations
  // compute gradients over batch for policy, Lagrangian multipliers and Q-function
  $\delta_\theta \leftarrow -\nabla_\theta \sum_{h_t \in \tau} \sum_j [\exp(Q_\phi(s_t, a_j, o_j)/\eta) \log \pi_\theta(a_j, o_j|h_t)]$ following Eq. 7 and 8
  $\delta_\eta \leftarrow \nabla_\eta g(\eta) = \nabla_\eta \eta\epsilon + \eta \sum_{h_t \in \tau} \log \sum_j [\exp(Q_\phi(s_t, a_j, o_j)/\eta)]$ following Eq. 15
  $\delta_\phi \leftarrow \nabla_\phi \sum_{(s_t, a_t, o_t) \in \tau} \left(Q_\phi(s_t, a_t, o_t) - Q_T\right)^2$ following Eq. 5
  update θ, η, ϕ // apply gradient updates
  if number of iterations = target update **then**
    $\pi' = \pi_\theta, Q' = Q_\phi$ // update target networks for policy π' and value function Q'

(Figure 2) with a third dimension representing the number of switches between options. Practically, this means that we are modelling $\pi^H(o_t, n_t|h_t)$ where n_t represents the number of switches until timestep t. We can now marginalize over *all* numbers of switches to again determine the option probabilities. Instead, to encourage option consistency across timesteps, we can sum over *only a subset* of nodes for all $n \leq N$ with N smaller than the maximal number of switches leading to $\pi^H(o_t|h_t) = \sum_{n_t=0}^{N} \pi^H(o_t, n_t|h_t)$.

For the first timestep, only 0 switches are possible, such that $\pi^H(o_0, n_0 = 0|h_0) = \pi^C(o_0|s_0)$ and 0 for all other values of n. For further timesteps, all edges resulting in option terminations β lead to the next step's option probabilities with increased number of switches $n_{t+1} = n_t + 1$. All edges representing the continuation of an option lead to $n_{t+1} = n_t$. This results in the computation of the joint distribution for $t > 0$:

$$\tilde{\pi}^H(o_t, n_t|h_t) = \sum_{\substack{o_{t-1}=1,\\ n_{t-1}=1}}^{M,N} p(o_t, n_t|s_t, o_{t-1}, n_{t-1}) \pi^H(o_{t-1}, n_{t-1}|h_{t-1}) \pi^L(a_{t-1}|s_{t-1}, o_{t-1}) \tag{9}$$

which can then be normalized using $\pi^H(o_t, n_t|h_t) = \tilde{\pi}^H(o_t, n_t|h_t) / \sum_{o'_t=1}^{M} \sum_{n'_t=1}^{L} \tilde{\pi}^H(o'_t, n'_t|h_t)$. The option and switch index transitions $p(o_t, n_t|s_t, o_{t-1}, n_{t-1})$ are further described in Equation 17 in the Appendix.

3. Experiments

In this section, we aim to answer a set of questions to better understand the contribution of different aspects to the performance of option learning - in particular with respect to the proposed method, HO2. To start, in Section 3.1 we explore two questions: (1) How well does HO2 perform in comparison to existing option learning methods? and (2) How important is off-policy training in this context? We use a set of common OpenAI gym (Brockman et al., 2016) benchmarks to answer these questions. In Section 3.2 we ask: (3) How do action abstraction in mixture policies and the additional temporal abstraction brought by option policies individually impact performance? We use more complex, pixel-based 3D robotic manipulation experiments to investigate these two aspects and evaluate scalability with respect to higher dimensional input and state spaces. We also explore: (4) How does increased temporal consistency impact performance, particularly with respect to sequential transfer with pre-trained options? Finally, we perform additional ablations in Section 3.3 to investigate the challenges of robust off-policy option learning and improve understanding of how options decompose behavior based on various environment and algorithmic aspects.

Across domains, we use HO2 to train option policies, RHPO (Wulfmeier et al., 2020) for the reduced case of mixture-of-Gaussians policies with sampling of options at every timestep and MPO (Abdolmaleki et al., 2018a) to train indi-

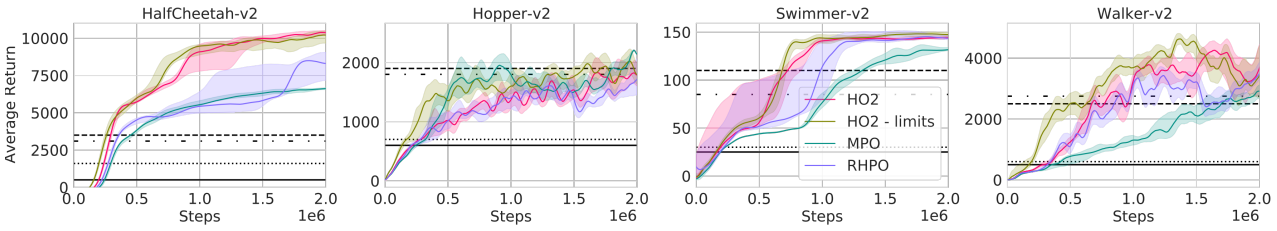

Figure 3: Results on OpenAI gym. Dashed black line represents DAC (Zhang & Whiteson, 2019), dotted line represents Option-Critic (Bacon et al., 2017), solid line represents IOPG (Smith et al., 2018), dash-dotted line represents PPO (Schulman et al., 2017) (approximate results after 2×10^6 steps from (Zhang & Whiteson, 2019)). We limit the number of switches to 5 for HO2-limits. HO2 consistently performs better or on par with existing option learning algorithms.

vidual flat Gaussian policies - all based on critic-weighted maximum likelihood estimation for policy optimization.

3.1. Comparison of Option Learning Methods

We compare HO2 (with and without limits on option switches) against competitive baselines for option learning in common, feature-based continuous action space domains. HO2 outperforms baselines including Double Actor-Critic (DAC) (Zhang & Whiteson, 2019), Inferred Option Policy Gradient (IOPG) (Smith et al., 2018) and Option-Critic (OC) (Bacon et al., 2017). With PPO (Schulman et al., 2017), we include a commonly used on-policy method for flat policies which in addition serves as the foundation for the DAC algorithm.

As demonstrated in Figure 3, HO2 performs better than or commensurate to existing option learning algorithms such as DAC, IOPG and Option-Critic as well as PPO. Training mixture policies (via RHPO (Wulfmeier et al., 2020)) without temporal abstraction slightly reduces both performance and sample efficiency but still outperforms on-policy methods in many cases. Here, enabling temporal abstraction (even without explicitly maximizing it) provides an inductive bias to reduce the search space for the high-level controller and can lead to more data-efficient learning, such that HO2 even without constraints performs better than RHPO.

Finally, while less data-efficient than HO2, even off-policy learning alone with flat Gaussian policies (here MPO (Abdolmaleki et al., 2018b)) can outperform current on-policy option algorithms, for example in the HalfCheetah and Swimmer domains while otherwise at least performing on par. This emphasizes the importance of a strong underlying policy optimization method.

Using the switch constraints for increasing temporal abstraction from Section 2 can provide minor benefits in some tasks but has overall only a minor effect on performance. We further investigate this setting in sequential transfer in Section 3.2. It has to be noted that given the comparable simplicity of these tasks, considerable performance gains are achieved with pure off-policy training. In the next section, we study more complex domains to isolate additional gains from action and temporal abstraction.

3.2. Ablations: Action Abstraction and Temporal Abstraction

We next ablate different aspects of HO2 on more complex simulated 3D robot manipulation tasks - stacking and the more dynamic ball-in-cup (BIC) - as displayed in Figure 4, based on robot proprioception and raw pixel inputs (64x64 pixel, 2 cameras for BIC and 3 for stacking). Since the performance of HO2 for training from scratch is relatively independent of switch constraints (Figure 3), we will simplify our figures by focusing on the base method. To reduce data requirements, we use a set of common techniques to improve data-efficiency and accelerate learning for all methods. We will apply multi-task learning with a related set of tasks to provide a curriculum, with details in Appendix A. Furthermore, we assign rewards for all tasks to any generated transition data in hindsight to improve data efficiency and exploration (Andrychowicz et al., 2017; Riedmiller et al., 2018; Wulfmeier et al., 2020; Cabi et al., 2017).

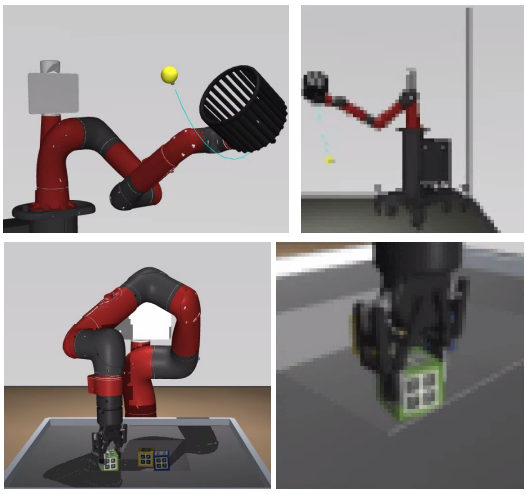

Figure 4: Ball-In-Cup (top) and Stacking (bottom). Left: Environments. Right: Example agent observations.

Across all tasks, except for simple positioning and reach tasks (see Appendix A), action abstraction improves performance (mixture policies via RHPO versus flat Gaussian policies via MPO). In particular the more challenging stacking tasks shown in Figure 5 intuitively benefit from shared sub-behaviors with easier tasks. Finally, the introduction of temporal abstraction (option policies via HO2 vs mixture policy via RHPO) further improves both performance and sample efficiency, especially on the more complex stacking tasks. The ability to learn explicit termination conditions, which can be understood as classifiers between two conditions, instead of the high-level controller, as classifier between all options, can considerably simplify the learning problem.

Optimizing for Temporal Abstraction There is a difference between simplifying the representation of temporal abstraction for the agent and explicitly maximizing it. The ability to represent temporally abstract behavior in HO2 via the use of explicit termination conditions consistently helps in prior experiments. However, these experiments show limited benefit when increasing temporal consistency (by limiting the number of switches following Section 2.3) for training from scratch.

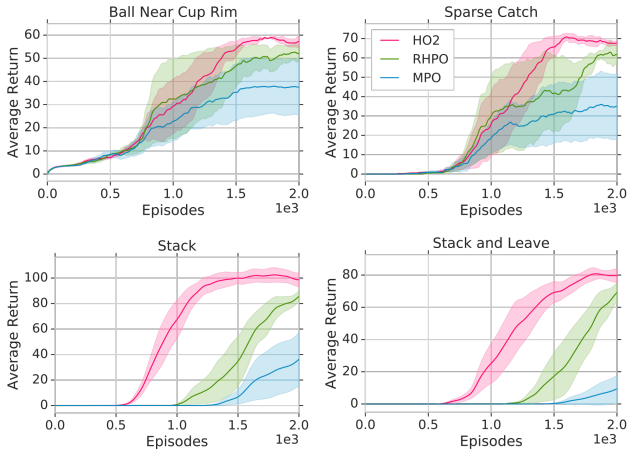

Figure 5: Results for option policies, and ablations via mixture policies, and single Gaussian policies (respectively HO2, RHPO, and MPO) with pixel-based ball-in-cup (left) and pixel-based block stacking (right). All four tasks displayed use sparse binary rewards, such that the obtained reward represents the number of timesteps where the corresponding condition - such as the ball is in the cup - is fulfilled. See Appendix B for details and additional tasks.

In this section, we further evaluate temporal abstraction for sequential transfer with pre-trained options. We first train low-level options for all tasks except for the most complex task in each domain by applying HO2. Next, given a set of pre-trained options, we only train the final task and compare training with and without maximizing temporal abstraction. We use the domains from Section 3.2, block stacking and BIC.As shown in Figure 6, we can see that more consistent options lead to increased performance in the transfer domain. Intuitively, increased temporal consistency and fewer switches lead to a smaller search space from the perspective of the high-level controller.

While the same mechanism should also apply for training from scratch, we hypothesize that the added complexity of simultaneously learning the low-level behaviors (while maximizing temporal consistency) outweighs the benefits. Finding a set of options which not only solve a task but also represent temporally consistent behavior can be harder, and require more data, than just solving the task.

3.3. Ablations: Off-policy, Robustness, Decomposition

In this section, we investigate different algorithmic aspects to get a better understanding of the method, properties of the learned options, and how to achieve robust training in the off-policy setting.

Off-policy Option Learning In off-policy hierarchical RL, the low-level policy underlying an option can change after trajectories are generated. This results in a non-

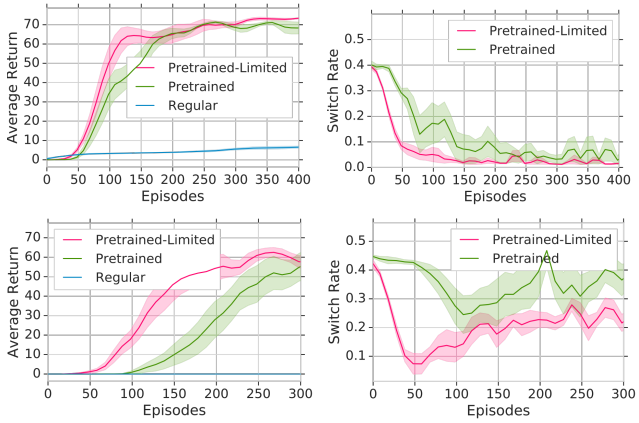

Figure 6: The sequential transfer experiments for temporal abstraction show considerable improvements for limited switches. Top: BIC. Bottom: Stack. In addition, we visualize the actual agent option switch rate in the environment to directly demonstrate the constraint's effect.

stationarity for training the high-level controller. In addition, including previously executed actions in the forward computation for component probabilities can introduce additional variance into the objective. In practice, we find that removing the conditioning on low-level probabilities (the π_L terms in Equation 3) improves performance and stability. The effect is displayed in Figure 7, where the conditioning of high-level component probabilities on low-level action probabilities (see Section 2) is detrimental.

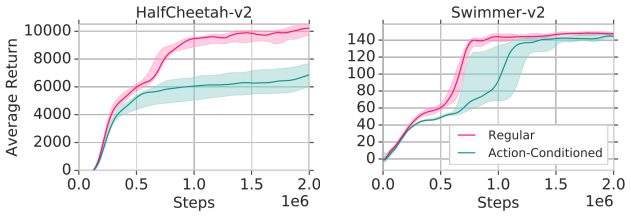

Figure 7: Results on OpenAI gym with/without option probabilities being conditioned on past actions.

We additionally evaluate this effect in the on-policy setting in Appendix A.4 and find its impact to be diminished, demonstrating the connection between the effect and an off-policy setting. While we apply this simple heuristic for HO2, the problem has lead to various off-policy corrections for goal-based HRL (Nachum et al., 2018b; Levy et al., 2017) which provide a valuable direction for future work.

Trust-regions and Robustness Previous work has shown the benefits of applying trust-region constraints for policy updates of non-hierarchical policies (Schulman et al., 2015; Abdolmaleki et al., 2018b). In this section, we vary the

strength of constraints on the option probability updates (both termination conditions β and the high-level controller π_C). As displayed in Figure 8, the approach is robust across multiple orders of magnitude, but very weak or strong constraints can considerably degrade performance. Note that a high value is essentially equal to not using a constraint and causes very low performance. Therefore, option learning here relies strongly on trust-region constraints. Further experiments can be found in Appendix A.5.

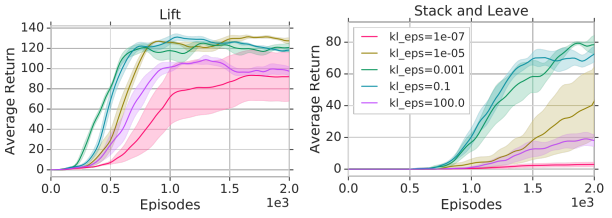

Figure 8: Block stacking results for two tasks with different trust-region constraints. Note that the importance of constraints increases for more complex tasks.

Decomposition and Option Clustering To investigate how HO2 uses its capacity and decomposes behavior into options, we apply it to a variety of simple and interpretable locomotion tasks. In these tasks, the agent body ("Ball", "Ant", or "Quadruped") must go to one of three targets in a room, with the task specified by the target locations and a selected target index. As shown for the "Ant" case in Figure 9, we find that option decomposition intuitively depends on both the task properties and algorithm settings. In particular *information asymmetry (IA)*, achieved by providing task information only to the high-level controller, can address degenerate solutions and lead to increased diversity with respect to options (as shown by the histogram over options) and more specialized options (represented by the clearer clustering of samples in action space). We can measure this quantitatively, using (1) the Silhouette score, a measure of clustering accuracy based on inter- and intra-cluster distances[1]; and (2) entropy over the option histogram, to quantify diversity. These metrics are reported in Table 1 for all bodies, with and without information asymmetry. The results show that for all cases, IA leads to greater option diversity and clearer separation of option clusters with respect to action space, state space, and task.

More extensive experiments and discussion can be found in Appendix A.1, including additional quantitative and qualitative results for the other bodies and scenarios. To summarize, the analyses yield a number of relevant observations, showing that (1) for simpler bodies like a "Ball", the options are interpretable (forward torque, and turning left/right at different rates); and (2) applying the switch constraint introduced in Section 3.2 leads to increased temporal abstraction without reducing the agent's ability to solve the task.

[1]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.

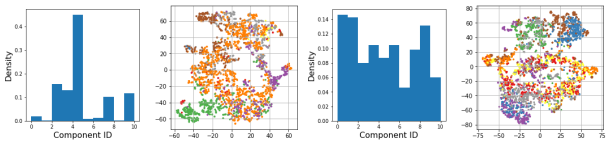

Figure 9: Analysis on Ant locomotion tasks, showing histogram over options, and t-SNE scatter plots in action space colored by option. Left: without IA. Right: with IA. Agents with IA use more components and show clearer option clustering in the action space.

Scenario		Option entropy	s (action)	s (state)	s (task)
Ball	No IA	1.80 ± 0.21	-0.30 ± 0.04	-0.25 ± 0.14	-0.13 ± 0.05
	With IA	2.23 ± 0.03	-0.13 ± 0.04	-0.11 ± 0.04	-0.05 ± 0.00
Ant	No IA	1.60 ± 0.08	-0.12 ± 0.02	-0.15 ± 0.07	-0.08 ± 0.03
	With IA	2.22 ± 0.04	-0.05 ± 0.01	-0.05 ± 0.01	-0.05 ± 0.01
Quad	No IA	1.55 ± 0.29	-0.07 ± 0.04	-0.12 ± 0.03	-0.11 ± 0.02
	With IA	2.23 ± 0.04	-0.03 ± 0.03	-0.03 ± 0.00	-0.05 ± 0.01

Table 1: Quantitative results indicating the diversity of options used (entropy), and clustering accuracy in action and state spaces (silhouette score s), with and without information asymmetry (IA). Switching constraints are applied in all cases. Higher values indicate greater separability by option / component.

4. Related Work

Hierarchy has been investigated in many forms in reinforcement learning to improve data gathering as well as data fitting aspects. Goal-based approaches commonly define a grounded interface between high- and low-level policies. The high level acts by providing goals to the low level, which is trained to achieve these goals (Dayan & Hinton, 1993; Levy et al., 2017; Nachum et al., 2018a;b; Vezhnevets et al., 2017), effectively generating separate objectives and improving exploration. These methods have been able to overcome very sparse reward domains but commonly require domain knowledge to define the interface. In addition, a hand-crafted interface can limit expressiveness of achievable behaviors.

Non-crafted, emergent interfaces within policies have been investigated from an RL-as-inference perspective via policies with continuous latent variables (Haarnoja et al., 2018; Hausman et al., 2018; Heess et al., 2016; Igl et al., 2019; Tirumala et al., 2019; Teh et al., 2017). Related to these

approaches, we provide a probabilistic inference perspective to off-policy option learning and benefit from efficient dynamic programming inference procedures. We furthermore build on the related idea of information asymmetry (Pinto et al., 2017; Galashov et al., 2018; Tirumala et al., 2019) - providing a part of the observations only to a part of the model. The asymmetry can lead to an information bottleneck affecting the properties of learned low-level policies. We build on the intuition and demonstrate how option diversity can be affected in ablations in Section 3.3.

At its core, our work builds on and investigates the option framework (Precup, 2000; Sutton et al., 1999), which can be seen as describing policies with an autoregressive, discrete latent space. Option policies commonly use a high-level controller to choose from a set of options or skills. These options additionally include termination conditions to enable a skill to represent temporally extended behavior. Without termination conditions, options can be seen as equivalent to components under a mixture distribution, and this simplified formulation has been applied successfully in different methods (Agostini & Celaya, 2010; Daniel et al., 2016; Wulfmeier et al., 2020).

Recent work has also investigated temporally extended low-level behaviours of fixed length (Frans et al., 2018; Li et al., 2019; Nachum et al., 2018b), which do not learn the option duration or termination condition. With HO2, enabling to optimize the extension of low-level behaviour in the option framework provides additional flexibility and removes the engineering effort of choosing the right hyperparameters.

The option framework has been further extended and improved for more practical application (Bacon et al., 2017; Harb et al., 2018; Harutyunyan et al., 2019; Precup et al., 2006; Riemer et al., 2018; Smith et al., 2018). HO2 relies on off-policy training and treats options as latent variables. This enables backpropagation through the option inference procedure and considerable improvements in comparison to efficient than approaches relying on on-policy updates and on-option learning purely for executed options. Related, IOPG (Smith et al., 2018) also considers an inference perspective but only includes on-policy results which naturally have poorer data efficiency. Finally, the benefits of options and other modular policy styles have also been applied in the supervised case for learning from demonstration (Fox et al., 2017; Krishnan et al., 2017; Shiarlis et al., 2018).

One important step to increase the robustness of option learning has been taken in (Zhang & Whiteson, 2019) by building on robust (on-policy) policy optimization with PPO (Schulman et al., 2017). HO2 has similar robustness benefits, but additionally improves data-efficiency by building on off-policy learning, hindsight inference of options, and additional trust-region constraints (Abdolmaleki et al., 2018b; Wulfmeier et al., 2020). Related inference procedures have also been investigated in imitation learning (Shiarlis et al., 2018) as well as on-policy RL (Smith et al., 2018).

In addition to inferring options in hindsight, off-policy learning enables us to assign rewards for multiple tasks, which has been successfully applied with flat, non-hierarchical policies (Andrychowicz et al., 2017; Riedmiller et al., 2018; Cabi et al., 2017) and goal-based hierarchical approaches (Levy et al., 2017; Nachum et al., 2018b).

5. Conclusions

We introduce a robust, efficient algorithm for off-policy training of option policies. The approach outperforms recent work in option learning on common benchmarks and is able to solve complex, simulated robot manipulation tasks from raw pixel inputs more reliably than competitive baselines. HO2 takes a probabilistic inference perspective to option learning, infers option and action probabilities for trajectories in hindsight, and performs critic-weighted maximum-likelihood estimation by backpropagating through the inference step. Being able to infer options for a given trajectory allows robust off-policy training and determination of updates for all instead of only for the executed options. It also makes it possible to impose constraints on the termination frequency independently of an environment's reward scale.

We separately analyze the impact of action abstraction (via mixture policies), and temporal abstraction (via options). We find that each abstraction independently improves performance. Additional maximization of temporal consistency for option choices is beneficial when transferring pre-trained options but displays a limited effect when learning from scratch. Furthermore, we investigate the consequences of the off-policyness of training data and demonstrate the benefits of trust-region constraints for option learning. We examine the impact of different agent and environment properties (such as information asymmetry, tasks, and embodiments) with respect to task decomposition and option clustering; a direction which provides opportunities for further investigation in the future. Finally, since our method is based on (weighted) maximum likelihood estimation, it can be adapted naturally to learn structured behavior representations in mixed data regimes, e.g. to learn from combinations of demonstrations, logged data, and online trajectories. This opens up promising directions for future work.

Acknowledgments

The authors would like to thank Peter Humphreys, Satinder Baveja, Tobias Springenberg, and Yusuf Aytar for helpful discussion and relevant feedback which helped to shape the publication. We additionally like to acknowledge the support of the DeepMind robotics lab for infrastructure and engineering support.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pp. 265–283, 2016.

Abdolmaleki, A., Springenberg, J. T., Degrave, J., Bohez, S., Tassa, Y., Belov, D., Heess, N., and Riedmiller, M. Relative entropy regularized policy iteration. *arXiv preprint arXiv:1812.02256*, 2018a.

Abdolmaleki, A., Springenberg, J. T., Tassa, Y., Munos, R., Heess, N., and Riedmiller, M. A. Maximum a posteriori policy optimisation. *CoRR*, abs/1806.06920, 2018b.

Agostini, A. and Celaya, E. Reinforcement learning with a gaussian mixture model. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2010.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., and Zaremba, W. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pp. 5048–5058, 2017.

Bacon, P.-L., Harb, J., and Precup, D. The option-critic architecture. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *ArXiv*, abs/1606.01540, 2016.

Cabi, S., Colmenarejo, S. G., Hoffman, M. W., Denil, M., Wang, Z., and Freitas, N. The intentional unintentional agent: Learning to solve many continuous control tasks simultaneously. In *Conference on Robot Learning*, pp. 207–216. PMLR, 2017.

Daniel, C., Neumann, G., Kroemer, O., and Peters, J. Hierarchical relative entropy policy search. *The Journal of Machine Learning Research*, 17(1):3190–3239, 2016.

Dayan, P. and Hinton, G. E. Feudal reinforcement learning. In *Advances in neural information processing systems*, pp. 271–278, 1993.

Fox, R., Krishnan, S., Stoica, I., and Goldberg, K. Multi-level discovery of deep options. *arXiv preprint arXiv:1703.08294*, 2017.

Frans, K., Ho, J., Chen, X., Abbeel, P., and Schulman, J. Meta learning shared hierarchies. In *International Conference on Learning Representations*, 2018.

Galashov, A., Jayakumar, S. M., Hasenclever, L., Tirumala, D., Schwarz, J., Desjardins, G., Czarnecki, W. M., Teh, Y. W., Pascanu, R., and Heess, N. Information asymmetry in kl-regularized rl. 2018.

Haarnoja, T., Hartikainen, K., Abbeel, P., and Levine, S. Latent space policies for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pp. 1846–1855, 2018.

Harb, J., Bacon, P.-L., Klissarov, M., and Precup, D. When waiting is not an option: Learning options with a deliberation cost. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Harutyunyan, A., Dabney, W., Borsa, D., Heess, N., Munos, R., and Precup, D. The termination critic. *CoRR*, abs/1902.09996, 2019. URL http://arxiv.org/abs/1902.09996.

Hausman, K., Springenberg, J. T., Wang, Z., Heess, N., and Riedmiller, M. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018.

Heess, N., Wayne, G., Tassa, Y., Lillicrap, T., Riedmiller, M., and Silver, D. Learning and transfer of modulated locomotor controllers. *arXiv preprint arXiv:1610.05182*, 2016.

Igl, M., Gambardella, A., Nardelli, N., Siddharth, N., Böhmer, W., and Whiteson, S. Multitask soft option learning. *arXiv preprint arXiv:1904.01033*, 2019.

Krishnan, S., Fox, R., Stoica, I., and Goldberg, K. Ddco: Discovery of deep continuous options for robot learning from demonstrations. *arXiv preprint arXiv:1710.05421*, 2017.

Levy, A., Konidaris, G., Platt, R., and Saenko, K. Learning multi-level hierarchies with hindsight. *arXiv preprint arXiv:1712.00948*, 2017.

Li, A. C., Florensa, C., Clavera, I., and Abbeel, P. Sub-policy adaptation for hierarchical reinforcement learning. *arXiv preprint arXiv:1906.05862*, 2019.

Nachum, O., Gu, S., Lee, H., and Levine, S. Near-optimal representation learning for hierarchical reinforcement learning. *arXiv preprint arXiv:1810.01257*, 2018a.

Nachum, O., Gu, S. S., Lee, H., and Levine, S. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 3303–3313, 2018b.

OpenAI, Andrychowicz, M., Baker, B., Chociej, M., Józefowicz, R., McGrew, B., Pachocki, J. W., Pachocki, J.,

Petron, A., Plappert, M., Powell, G., Ray, A., Schneider, J., Sidor, S., Tobin, J., Welinder, P., Weng, L., and Zaremba, W. Learning dexterous in-hand manipulation. *CoRR*, abs/1808.00177, 2018. URL http://arxiv.org/abs/1808.00177.

Pinto, L., Andrychowicz, M., Welinder, P., Zaremba, W., and Abbeel, P. Asymmetric actor critic for image-based robot learning. *arXiv preprint arXiv:1710.06542*, 2017.

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

Precup, D., Paduraru, C., Koop, A., Sutton, R. S., and Singh, S. P. Off-policy learning with options and recognizers. In Weiss, Y., Schölkopf, B., and Platt, J. C. (eds.), *Advances in Neural Information Processing Systems 18*, pp. 1097–1104. MIT Press, 2006.

Rabiner, L. R. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

Riedmiller, M., Hafner, R., Lampe, T., Neunert, M., Degrave, J., Van de Wiele, T., Mnih, V., Heess, N., and Springenberg, J. T. Learning by playing-solving sparse reward tasks from scratch. *arXiv preprint arXiv:1802.10567*, 2018.

Riemer, M., Liu, M., and Tesauro, G. Learning abstract options. In *Advances in Neural Information Processing Systems*, pp. 10424–10434, 2018.

Schulman, J., Levine, S., Moritz, P., Jordan, M., and Abbeel, P. Trust region policy optimization. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning-Volume 37*, pp. 1889–1897. JMLR. org, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shiarlis, K., Wulfmeier, M., Salter, S., Whiteson, S., and Posner, I. Taco: Learning task decomposition via temporal alignment for control. In *International Conference on Machine Learning*, pp. 4661–4670, 2018.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

Smith, M., Hoof, H., and Pineau, J. An inference-based policy gradient method for learning options. In *International Conference on Machine Learning*, pp. 4703–4712, 2018.

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

Teh, Y. W., Bapst, V., Czarnecki, W. M., Quan, J., Kirkpatrick, J., Hadsell, R., Heess, N., and Pascanu, R. Distral: Robust multitask reinforcement learning. *CoRR*, abs/1707.04175, 2017. URL http://arxiv.org/abs/1707.04175.

Tirumala, D., Noh, H., Galashov, A., Hasenclever, L., Ahuja, A., Wayne, G., Pascanu, R., Teh, Y. W., and Heess, N. Exploiting hierarchy for learning and transfer in kl-regularized rl. *arXiv preprint arXiv:1903.07438*, 2019.

Vezhnevets, A. S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., and Kavukcuoglu, K. Feudal networks for hierarchical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3540–3549. JMLR. org, 2017.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575 (7782):350–354, 2019.

Wulfmeier, M., Abdolmaleki, A., Hafner, R., Springenberg, J. T., Neunert, M., Siegel, N., Hertweck, T., Lampe, T., Heess, N., and Riedmiller, M. Compositional Transfer in Hierarchical Reinforcement Learning. In *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020. doi: 10.15607/RSS.2020.XVI.054.

Zhang, S. and Whiteson, S. Dac: The double actor-critic architecture for learning options. In *Advances in Neural Information Processing Systems*, pp. 2010–2020, 2019.