On the Role of Weight Sharing During Deep Option Learning

Matthew Riemer,[1] **Ignacio Cases,**[2] **Clemens Rosenbaum,**[3] **Miao Liu,**[1] **Gerald Tesauro**[1]
[1]IBM Research, Yorktown Heights, NY
[2]Linguistics Department and Stanford NLP Group, AI Lab, Stanford University
[3]College of Information and Computer Sciences, University of Massachusetts Amherst

Abstract

The options framework is a popular approach for building temporally extended actions in reinforcement learning. In particular, the option-critic architecture provides general purpose policy gradient theorems for learning actions from scratch that are extended in time. However, past work makes the key assumption that each of the components of option-critic has independent parameters. In this work we note that while this key assumption of the policy gradient theorems of option-critic holds in the tabular case, it is always violated in practice for the deep function approximation setting. We thus reconsider this assumption and consider more general extensions of option-critic and hierarchical option-critic training that optimize for the full architecture with each update. It turns out that not assuming parameter independence challenges a belief in prior work that training the policy over options can be disentangled from the dynamics of the underlying options. In fact, learning can be sped up by focusing the policy over options on states where options are actually likely to terminate. We put our new algorithms to the test in application to sample efficient learning of Atari games, and demonstrate significantly improved stability and faster convergence when learning long options. [1]

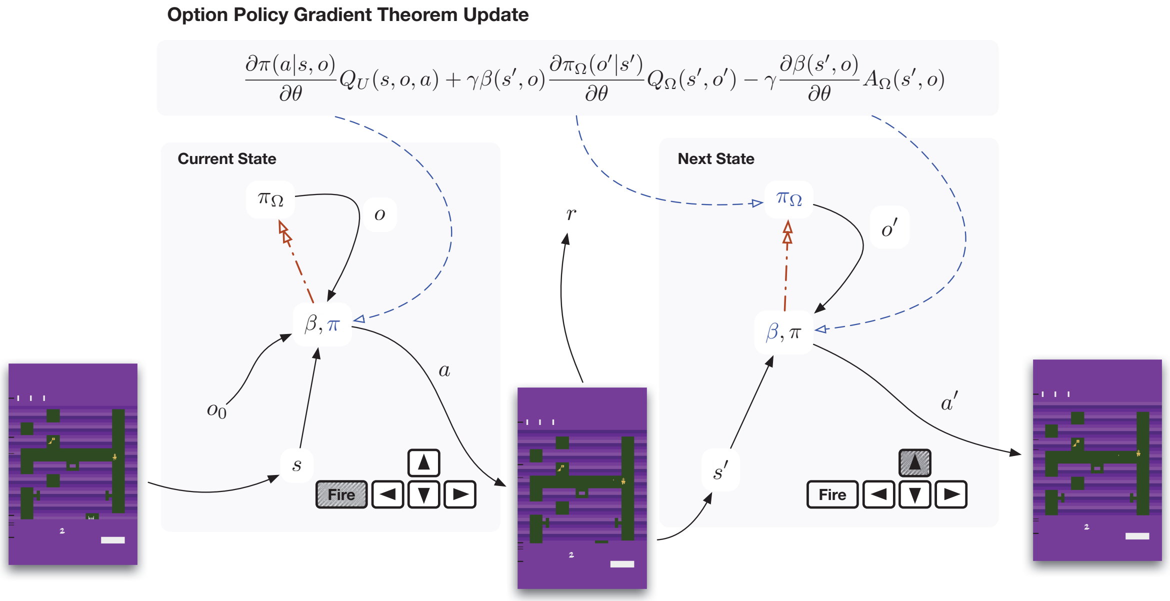
Introduction

Developing systems that can autonomously create temporal abstractions is a major problem in scaling deep reinforcement learning (RL). *Options* (Sutton, Precup, and Singh 1999; Precup 2000) provide a general purpose framework for defining temporally abstract courses of action for learning and planning in RL. This is a very promising direction with the potential to allow for more coherent exploration and improved long term credit assignment by effectively pruning the number of decision nodes. The popular option-critic (Bacon, Harb, and Precup 2017) learning framework blurs the line between option discovery and option learning. These approaches have achieved success when applied to Q-learning on Atari (Bacon, Harb, and Precup 2017), but also with continuous action spaces (Klissarov et al. 2017) and asynchronous parallelization (Harb et al. 2017). Additionally, this framework was recently extended to the hierarchical option-critic framework (Riemer, Liu, and Tesauro 2018), which allows networks to learn an arbitrary depth hierarchy of high level (longer) options and low level (shorter) options.

With the recent successes of Deep Learning based function approximation applied to RL, a major point of interest has become the derivation of theoretically justified policy gradient theorems. Policy gradient theorems (originally developed for learning with primitive actions by Sutton et al. (2000)) are critical to the success of Deep RL as they define the gradient steps that update the parameters of deep neural networks to maximize the expected reward with gradient descent style learning rules. The option-critic framework (Bacon, Harb, and Precup 2017) has garnered significant interest largely because it defined the first policy gradient theorems that can be used to update a neural network architecture that is endowed with options that are learned from scratch along with the network. The high level role of options as a temporally extended form of actions is clear. However, how these abstract actions should be composed with respect to the parameters of a neural network is far less clear and has not been closely studied. In fact, the default architecture framework for learning options (Bacon, Harb, and Precup 2017; Riemer, Liu, and Tesauro 2018) functions in a setting where the underlying assumptions of existing algorithms for optimizing these architectures do not hold.

Past work on deriving policy gradient theorems for option models (Bacon, Harb, and Precup 2017) has assumed that the parameters of the option policies $\boldsymbol{\pi}$, the policy over options $\boldsymbol{\pi_\Omega}$ and the termination functions $\boldsymbol{\beta}$ are independent of each other (i.e. $\theta_{\boldsymbol{\pi}} \cap \theta_{\boldsymbol{\pi_\Omega}} \cap \theta_{\boldsymbol{\beta}} = \emptyset$). This assumption gets even more strict for work on hierarchical option models where policies and termination functions are assumed to have independent parameters at each level of the hierarchy as well (Riemer, Liu, and Tesauro 2018). In practice, while this may hold for tabular problems, this has never been true in application to deep neural networks. In contrast, most parameters have been totally shared using a shared feature extraction network common across all model components and a private output layer for each component. This setup closely follows conventions from multi-task learning (Caruana 1997) with neural networks in the supervised setting. Nonetheless, the policy gradient theorems used to optimize these architectures

[1]See our arxiv version for the appendix and additional details.



Option Policy Gradient Theorem Update
$\frac{\partial \pi(a|s,o)}{\partial \theta} Q_U(s,o,a) + \gamma\beta(s',o)\frac{\partial \pi_\Omega(o'|s')}{\partial \theta} Q_\Omega(s',o') - \gamma\frac{\partial \beta(s',o)}{\partial \theta} A_\Omega(s',o)$
Current State
π_Ω
o
β, π
o_0
s
a
r
Fire
Next State
π_Ω
o'
β, π
s'
a'
Fire

Figure 1: We illustrate the training and execution of an option-critic architecture trained with the generalized option-critic policy gradient update rule on the game Tutankham. Blue lines represent gradients and red lines represent option termination.

have not actually been valid.

In this work, we seek to provide a remedy to this miss-match between the architectures used for deep option learning and the policy gradient theorems used to provide their learning rule. We achieve this by assuming in our derivations that all parameters are shared throughout all components of our model rather than assuming no sharing. This assumption is more general because even if a model component does not use certain parameters, it is valid to optimize for all parameters across all components because each component will only be influenced by the parameters that it uses. To be more concrete, if we assume some global set of shared parameters $\boldsymbol{\theta}$ such that $\theta_{\boldsymbol{\pi}} \in \boldsymbol{\theta}$, $\theta_{\boldsymbol{\pi}_\Omega} \in \boldsymbol{\theta}$, and $\theta_{\boldsymbol{\beta}} \in \boldsymbol{\theta}$, we can say for example that the set of parameters $\theta_{\boldsymbol{\pi}*} = \boldsymbol{\theta} - \theta_{\boldsymbol{\pi}}$ induces a zero gradient for $\boldsymbol{\pi}$ i.e. $\frac{\partial \boldsymbol{\pi}}{\partial \theta_{\boldsymbol{\pi}*}} = 0$. Past work (Bacon, Harb, and Precup 2017; Riemer, Liu, and Tesauro 2018) has typically derived separate learning rules for each policy and termination function with respect to their own parameters. By assuming instead to have one shared set of parameters, we arrive at a single learning rule that optimizes for the entire system with respect to its parameters $\boldsymbol{\theta}$. We will thus call this architecture level learning rule the *option-critic policy gradient* (OCPG) or more generally the *hierarchical option-critic policy gradient* (HOCPG) when modeling deep hierarchies of options.

Our paper aims to shed light on the disconnect between theory and practice in deep option-critic learning and motivates the benefits of a corrected update rule for efficient learning of long options. Our experiments in challenging RL environments with high dimensional state spaces such as Atari demonstrate the benefits of OCPG and HOCPG when using typical strategies for weight sharing across option model components. Additionally, this result is a critical first step towards developing more sophisticated weight sharing schemes across deep option models. It is extremely important that option models allow for more flexibility in this regard as the current methodology (Riemer, Liu, and Tesauro 2018) cannot appropriately handle, for example, the case when the same low level option is called within different high level options.

Background and Notation

A Markov Decision Process (MDP) is defined with a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, a transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to (\mathcal{S} \to [0,1])$ and a reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. Following standard practice (Bacon, Harb, and Precup 2017), we develop our ideas assuming discrete state and action sets, while our results extend to continuous spaces using usual measure-theoretic assumptions. A policy is defined as a probability distribution over actions conditioned on states, $\pi : \mathcal{S} \times \mathcal{A} \to [0,1]$. The value function of a policy π is the expected return $V_\pi(s) = \mathbb{E}_\pi[\sum_{t=0}^\infty \gamma^t r_{t+1} | s_0 = s]$ with an action-value function of $Q_\pi(s,a) = \mathbb{E}_\pi[\sum_{t=0}^\infty \gamma^t r_{t+1} | s_0 = s, a_0 = a]$ where $\gamma \in [0,1)$ is the *discount factor*.

Policy gradient methods improve a policy by performing gradient ascent over a family of parameterized stochastic policies, π_θ. The policy gradient theorem (Sutton et al. 2000) provides an expression to compute the gradient of the discounted reward objective with respect to θ and a designated starting state s_0 in a straightforward expression. The theorem defines the gradient update as $\sum_s \mu_{\pi_\theta}(s|s_0) \sum_a \frac{\partial \pi_\theta(a|s)}{\partial \theta} Q_{\pi_\theta}(s,a)$. Here $\mu_{\pi_\theta}(s|s_0) = \sum_{t=0}^\infty \gamma^t P(s_t = s|s_0)$ is defined as the discounted weighting of the states along the trajectories starting from initial state s_0.

The options framework (Sutton, Precup, and Singh 1999; Precup 2000) formalizes temporally extended actions in RL.

A Markovian option $o \in \Omega$ is a triple (I_o, π_o, β_o) where $I_o \subseteq S$ represents an initiation set, π_o represents an intra-option policy, and $\beta_o : \mathcal{S} \to [0,1]$ represents a termination function. Many algorithms (such as option-critic) assume that all options are available everywhere, removing the need to explicitly model I_o. MDPs with options become SMDPs (Puterman 1994) with an optimal value function over options $V^*_\Omega(s)$ and option-value function $Q^*_\Omega(s,o)$.

The option-critic architecture (Bacon, Harb, and Precup 2017) leverages a call-and-return option execution model where an agent picks option o according to its policy over options $\pi_\Omega(o|s)$, then follows the intra-option policy $\pi(a|s,o)$ until termination (as determined by $\beta(s,o)$). Termination then triggers a repetition of this procedure. Let $\pi_{\theta_\pi}(a|s,o)$ denote the intra-option policy of option o parametrized by θ_π and $\beta_{\theta_\beta}(s,o)$ the termination function of o parameterized by θ_β. The option-value function is then defined as:

$$Q_\Omega(s,o) = \sum_a \pi_{\theta_\pi}(a|s,o) Q_U(s,o,a), \tag{1}$$

$Q_U : \mathcal{S} \times \Omega \times \mathcal{A} \to \mathbb{R}$ is defined as the value of an action given the context of a state-option pair:

$$Q_U(s,o,a) = r(s,a) + \gamma \sum_{s'} P(s'|s,a) U(s',o). \tag{2}$$

These (s,o) pairs define an augmented state space (Levy and Shimkin 2011). Instead, the option-critic architecture leverages the function $U : \Omega \times \mathcal{S} \to \mathbb{R}$ which is called the option-value function upon arrival (Sutton, Precup, and Singh 1999). The value of selecting option o upon entering s' is:

$$U(s',o) = (1 - \beta_{\theta_\beta}(s',o)) Q_\Omega(s',o) + \beta_{\theta_\beta}(s',o) V_\Omega(s'). \tag{3}$$

We adopt a notation for clarity where we omit θ_π and θ_β which Q_U and U both depend on. The *intra-option policy gradient theorem* results from taking the derivative of the expected discounted return with respect to the intra-option policy parameters θ_π, defining the update rule for π:

$$\sum_{s,o} \mu_\Omega(s,o|s_0,o_0) \sum_a \frac{\partial \pi_{\theta_\pi}(a|s,o)}{\partial \theta_\pi} Q_U(s,o,a). \tag{4}$$

μ_Ω is defined as the discounted weighting of (s,o) along trajectories originating from (s_0,o_0) : $\mu_\Omega(s,o|s_0,o_0) = \sum_{t=0}^\infty \gamma^t P(s_t = s, o_t = o|s_0,o_0)$. Likewise, the *termination gradient theorem* results from taking the derivative of the expected discounted return with respect to the termination policy parameters θ_β and defines the update rule for β with initial condition (s_1,o_0):

$$-\sum_{s',o} \mu_\Omega(s',o|s_1,o_o) \frac{\partial \beta_{\theta_\beta}(s',o)}{\partial \theta_\beta} A_\Omega(s',o), \tag{5}$$

where A_Ω is the advantage function over options, $A_\Omega(s',o) = Q_\Omega(s',o) - V_\Omega(s')$.

The hierarchical option-critic architecture (Riemer, Liu, and Tesauro 2018) extends option-critic models to an arbitrarily deep N level hierarchy of high level options and low level options below them. We adopt the notation from (Riemer, Liu, and Tesauro 2018) $x^{i:i+j} = x^i, ..., x^{i+j}$. This implies that $x^{i:i+j}$ denotes a list of variables in the range of i through $i+j$. In this hierarchical architecture, $\pi^1_{\theta^1}(o^1|s)$ is the policy over the most abstract options in the hierarchy o^1. Once o^1 is chosen, o^2 is chosen with policy $\pi^2_{\theta^2}(o^2|s,o^1)$, which is the next highest level policy. This process continues until reaching policy $\pi^N_{\theta^N}(a|s,o^{1:N-1})$. π^N is the lowest level policy and finally selects over the primitive action space. Each level of the option hierarchy also has a complimentary termination function $\beta^1_{\phi^1}(s,o^1), ..., \beta^{N-1}_{\phi^{N-1}}(s,o^{1:N-1})$. Termination is bottom up, so high level options can only terminate when all lower level options have terminated first.

At each level of abstraction ℓ, the hierarchical option-critic architecture has an analogous option-value function $Q_\Omega(s,o^{1:\ell})$, value of selecting an option in the presence of previously selected options $Q_U(s,o^{1:\ell})$, and option-value function upon arrival $U(s,o^{1:\ell})$. The *hierarchical intra-option policy gradient theorem* results from taking the derivative of the expected discounted return with respect to the policy parameters θ^ℓ, defining the update rule for π^ℓ:

$$\sum_{s,o^{1:\ell}} \mu_\Omega(s,o^{1:\ell}|s_0,o_0^{1:N-1}) \frac{\partial \pi^\ell_{\theta^\ell}(o^\ell|s,o^{1:\ell-1})}{\partial \theta^\ell} Q_U(s,o^{1:\ell}), \tag{6}$$

where μ_Ω is defined as the discounted weighting of $(s,o^{1:\ell})$ along trajectories originating from $(s_0,o_0^{1:\ell})$: $\mu_\Omega(s,o^{1:\ell}|s_0,o_0^{1:N-1}) = \sum_{t=0}^\infty \gamma^t P(s_t = s, o_t^{1:\ell} = o^{1:\ell}|s_0,o_0^{1:N-1})$. The *hierarchical termination gradient theorem* results from taking the derivative of the expected discounted return with respect to the termination policy parameters ϕ^ℓ and defines the update rule for β^ℓ for the initial condition $(s_1,o_0^{1:N-1})$:

$$-\sum_{s,o^{1:\ell}} \mu_\Omega(s,o^{1:\ell}|s_1,o_0^{1:N-1}) \prod_{i=\ell+1}^{N-1} \beta^i_{\phi^i}(s,o^{1:i}) \Big(\frac{\partial \beta^\ell_{\phi^\ell}(s,o^{1:\ell})}{\partial \phi^\ell} A_\Omega(s,o^{1:\ell}) \Big), \tag{7}$$

where A_Ω is the advantage function over a hierarchy options.

Policy Gradient Theorems Over A Full Option-Critic Architecture

We now turn our attention to deriving policy gradient theorems for option-critic and hierarchical option-critic models by taking the derivative of the expected return with respect to a global set of shared parameters $\boldsymbol{\theta}$ rather than individually for each component at each level of abstraction.

Option-Critic Policy Gradients

For the standard option-critic model, we follow the original paper and take the derivative of $Q_\Omega(s,o)$, but now with respect to $\boldsymbol{\theta}$ rather than the parameters of the different system components. This can be done by substituting in equation 1.

Lemma 1 (Option-Critic Policy Gradients). *Given a set of Markov options with stochastic policies and termination functions differentiable in their parameters $\boldsymbol{\theta}$ governing each option policy π, termination function β, and the policy over options π_Ω, the gradient of the expected discounted return with respect to $\boldsymbol{\theta}$ and initial conditions (s_0, o_0) is:*

$$\sum_{s,o,s'} \mu_\Omega(s,o,s'|s_0,o_0)\Big(\sum_a \frac{\partial \pi(a|s,o)}{\partial \boldsymbol{\theta}} Q_U(s,o,a)$$
$$+ \sum_{o'} \gamma \beta(s',o) \frac{\partial \pi_\Omega(o'|s')}{\partial \boldsymbol{\theta}} Q_\Omega(s',o') - \gamma \frac{\partial \beta(s',o)}{\partial \boldsymbol{\theta}} A_\Omega(s',o)\Big),$$

where μ_Ω is a discounted weighting of augmented state tuples along trajectories starting from (s_0, o_0) : $\mu_\Omega(s,o,s'|s_0,o_0) = \sum_{t=0}^{\infty} \gamma^t P(s_t = s, o_t = o, s_{t+1} = s'|s_0,o_0)$. We provide one proof in the appendix in the style of (Bacon, Harb, and Precup 2017) and another complementary proof following (Kostas, Nota, and Thomas 2019). An important point we are making here is that all option-critic style policy gradient theorems are just special cases of the more general co-agent policy gradient theorem (Kostas, Nota, and Thomas 2019). In the appendix we also provide a formal A3C style algorithm and computational analysis. Despite a more complex update, we theoretically and empirically show very similar computation time to prior option-critic models.

Previously, Bacon (2018) considered an almost identical policy gradient termed the *joint gradient*. Bacon (2018) did not include the discount factor for the next state, and no similar algorithms have actually been implemented in prior work to the best of our knowledge. However, we consider this result a lemma towards developing a more general theorem. It is not on its own a major theoretical result because it already existed and was first established by Bacon (2018) and in a more general form by Kostas, Nota, and Thomas (2019). Our proof is not exactly the same as (Bacon 2018) mostly because it starts with a different equation for the value function.

It is interesting that when using $\boldsymbol{\theta}$ we see the emergence of a single update rule with three separate terms that closely parallel the individual option-critic update rules for π, π_Ω, and β. However, there are some considerable differences as well. There is now a formal acknowledgement of the policy over options and termination function being updated at the next state rather than the current state, which as such includes a multiplicative factor of γ. During training, in most cases this likely has a minor effect, especially if γ is close to 1.

The most influential new term is the multiplication by β that is present in the update rule for the policy over options. This multiplicative factor is quite intuitive as it modulates the importance of updates to the policy over options by the likelihood that the current option terminates and the policy over options is actually used. Here, β appears as a coupling factor between the dynamics of the policy over options and the dynamics of the underlying options being selected over. When the termination function is close to firing, the contribution of the update to the policy over options is the largest possible. In contrast, the smallest updates are obtained when the termination likelihood is close to zero. This term of the update therefore prioritizes likely transition points between options rather than treating all states equally. This is an interesting result as it shows that to strictly follow the exact policy gradient for the option-critic architecture, the policy over options cannot be viewed in total isolation from the underlying option dynamics as done in previous work by using simple Q-Learning or actor-critic. See Figure 2 for an illustration of how this refined policy gradient theorem promotes improved sample efficiency for learning the policy over options.

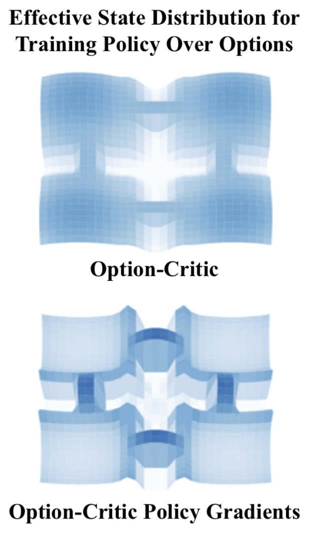


Figure 2: Illustration of effective state distribution for training π_Ω with OCPG in four rooms environment. *Top*: Typical option-critic learning focuses uniformly over states visited by the policy. *Bottom*: OCPG transforms the distribution to focus on the transition points where options are likely to terminate and the policy over options is actually used.

The last difference that it is important to mention is the inclusion of o' and not just o in the update rule. One could be concerned about adding to the transition tuple needed for updates. This is not an issue at all for straightforward application of the option-critic policy gradient to on-policy learning. However, it may not prove useful for off-policy updates where the current policy is significantly different from the policy used in the environment. In those cases, this term can be removed from transition tuple stored and recovered by sampling from the current policy over options. Additionally, even for the o term, it would probably be better to consider how the changing characteristics and current policy align with past behaviors as demonstrated by Nachum et al. (2018).

A Generalized Hierarchical Option-Critic Policy Gradient Theorem

In this section, following (Riemer, Liu, and Tesauro 2018), we seek to find a generalization of the option-critic policy gradient to an arbitrarily deep option hierarchy with N levels of abstraction. As we did in the last section, we start by taking the derivative of the option-value function for all active options $Q_\Omega(s, o^{1:N-1})$ with respect to $\boldsymbol{\theta}$. The derivation proceeds similarly to the one for the option-critic policy gradient. However, we now consider the complexities of an arbitrarily deep augmented state space of options.

Theorem 1 (Hierarchical Option-Critic Policy Gradient

Theorem). *Given an N level hierarchical set of Markov options with stochastic option policies at each level π^ℓ and termination functions at each level β^ℓ differentiable in their parameters $\boldsymbol{\theta}$, the gradient of the expected discounted return with respect to $\boldsymbol{\theta}$ and initial conditions $(s_0, o_0^{1:N-1})$ is:* [2]

$$\sum_{s,o^{1:N-1},s'} \mu_\Omega(s, o^{1:N-1}, s'|s_0, o_0^{1:N-1}) \Bigg(\sum_a \frac{\partial \pi(a|s, o^{1:N-1})}{\partial \boldsymbol{\theta}} Q_U(s, o^{1:N-1}, a) + \gamma \sum_{o'^{1:N-1}} \sum_{\ell=1}^{N-1} \prod_{k=N-1}^{\ell} \Bigg[\beta^k(s', o^{1:k}) \frac{\partial \pi^\ell(o'^\ell|s', o'^{1:\ell-1})}{\partial \boldsymbol{\theta}} Q_\Omega(s', o'^{1:\ell}) P_{\pi,\beta}(o'^{1:\ell-1}|s', o^{1:\ell-1}) \Bigg] - \gamma \sum_{\ell=1}^{N-1} \frac{\partial \beta^\ell(s', o^{1:\ell})}{\partial \boldsymbol{\theta}} A_\Omega(s', o^{1:\ell}) \prod_{k=N-1}^{\ell+1} \beta^k(s', o^{1:k}) \Bigg),$$

where μ_Ω is a discounted weighting of augmented state tuples along trajectories starting from $(s_0, o_0^{1:N-1})$: $\mu_\Omega(s, o^{1:N-1}, s'|s_0, o_0^{1:N-1}) = \sum_{t=0}^\infty \gamma^t P(s_t = s, o_t^{1:N-1} = o^{1:N-1}, s_{t+1} = s'|s_0, o_0^{1:N-1})$. $P_{\pi,\beta}(o'^{1:\ell-1}|s', o^{1:\ell-1})$ is the probability while at the next state and terminating the options for the current state that the agent arrives at a particular set of next option selections. We provide a proof in the appendix in the style of (Riemer, Liu, and Tesauro 2018) and another proof in the style of (Kostas, Nota, and Thomas 2019). We also provide a formal A3C style algorithm and analysis theoretically showing very similar computation per step to prior hierarchical option-critic models.

We first would like to point out that this theorem is a generalization of the option-critic policy gradient in the previous section. In fact, when $N = 2$ the hierarchical option-critic policy gradient theorem should be exactly the same. [3] In comparison to the original hierarchical intra-option policy gradient theorem and hierarchical termination gradient theorem from (Riemer, Liu, and Tesauro 2018), our new theorem has many of the same terms but with only one update rule rather than $2N - 1$ different update rules. The other chief differences with the original work are similar to the last section in that option policies and termination functions update with respect to the next state. Additionally, we again see the emergence of a dependence of the option policy update rules on the likelihood that the option policy is actually used. The option policy is only used if the both the option at that level of abstraction and all lower level options terminate.

Empirical Analysis

We now seek to evaluate how OCPG and HOCPG perform in a function approximation setting with a complex state space and thus consider the Atari games (Bellemare et al. 2013). We utilize the popular Open AI Gym environments for these games and use the default settings. We extend A3C from a popular PyTorch repository and provide further details on our setup in the appendix. Our architecture follows Mnih et al. (2015) consisting of a feature extractor common across all components of the architecture with 4 convolutional layers each followed by a max pooling and ReLU layer. This output is then fed into an LSTM as in (Mnih et al. 2016).

Implementation Details: We implement option-critic policy gradients (OCPG) using the variant of A2OC outlined in algorithm 1 of the appendix and implement the hierarchical option-critic policy gradients (HOCPG) following algorithm 2 of the appendix. Our primary baselines (OC) and (HOC) are standard version of A2OC and A2HOC respectively using the intra-option policy gradient theorem and termination gradient theorem for training. For easier direct comparison with our new full architecture level policy gradient theorem, we leverage actor-critic learning for the policy over options rather than arbitrarily using Q-Learning. This allows us to directly compare the ability of models to implement a policy gradient theorem for the full system. All of our models use 8 options following past work, and a learning rate of 1e-4. Following (Harb et al. 2017) we run 16 parallel asynchronously updating threads for each game. We also run one thread for evaluation that we do not learn from. We report the average and standard deviation of the reward for the most recent 150 evaluation episodes across ten runs. To ensure that our analysis is statistically sound, given the high variance that is typical for deep reinforcement learning, we follow best practices from (Colas, Sigaud, and Oudeyer 2018).

We should also note that it was the aim of prior work to study how and when options are useful. Our paper instead focuses on a particular systemic and problem agnostic aspect of the optimization of option-critic learning. As such, in the main text we only have space to report our main results. However, we have provided additional analysis of the options that are actually learned by each model and where potential learning advantages may be coming from in the appendix.

Settings Explored: It is important to note that Lemma 1 and Theorem 1 do not differ significantly from past work on option-critic and hierarchical option-critic learning for the corner case where all options always terminate. As such, it is clear that the gap with old approaches for option-critic learning becomes more substantial as options become more extended in time. On the other hand, this fact poses a challenge for conducting experiments that highlight the value gained as a result of the modified gradient. This is because option-critic style architectures since the first paper (Bacon, Harb, and Precup 2017) have needed to regularize the advantage function during the updates to the termination function using a parameter η to avoid trivially learning to always terminate. Option-critic style models paired with a regularizer can have the ability to learn options ranging the spectrum of temporal lengths, but have also been shown to be quite sensitive to the value of this parameter (Harb et al. 2017). In our experiments, we attempt to understand the role that regularization and option termination frequency have on the behavior of OCPG and OC. To do this we consider two settings of interest. In the first setting we consider a simple schedule (similar to a learning rate schedule) for η in which OCPG and HOCPG perform quite well. Then we explore choosing a reasonable fixed η that leads to options that are both extended in time and still divide the episode into segments.

[2] π^ℓ is expressed as a function $\pi^1(o^1|s)$ when $\ell = 1$.

[3] Note that $P_{\pi,\beta} = 1$ when $N = 2$.

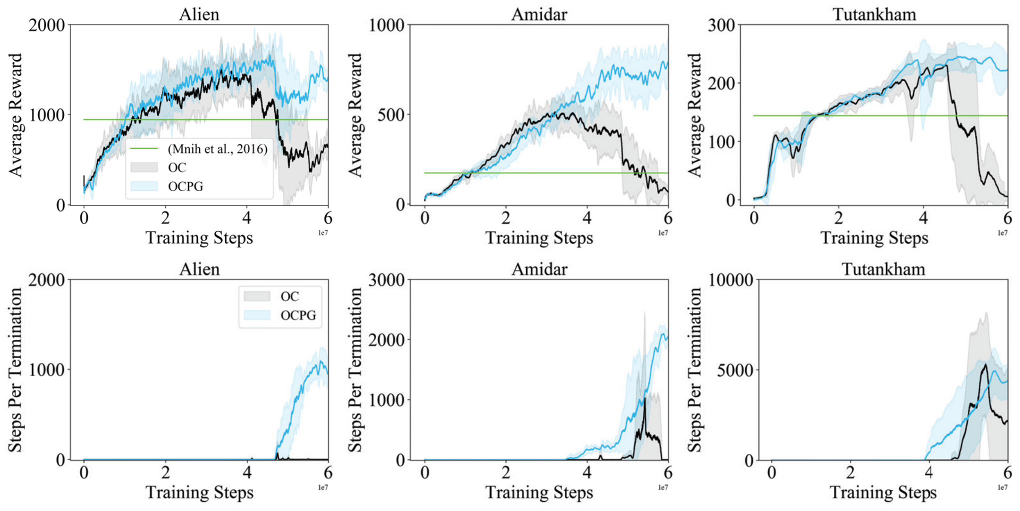


Figure 3: Option-critic style model performance and steps per termination during learning with a η schedule.

Regularization Schedule

While high values of η lead to options that are of longer duration, they have the drawback of leading to solutions that are merely η-optimal (Harb et al. 2017). This implies that we would only arrive at an optimal policy for the base MDP if $\eta = 0$. With this in mind, it seems quite natural to set a schedule for an agent trying to solve a task where first the agent focuses on learning skills from scratch without temporal abstraction and then the agent gradually learns to terminate less, decomposing the problem in time. Before any useful policies are learned, the regularization on the advantage may only serve to impede the proper convergence of the model. To test out this theory, we consider Atari experiments with training for 60 million steps while implementing a simple regularization schedule where η changes every 15 million steps. At the beginning we set $\eta = 0.0$ for 15 million steps before setting $\eta = 0.01$. Next, we set $\eta = 0.1$ at 30 million steps and $\eta = 1.0$ at 45 million steps. We tested six runs of each algorithm in this setup on three maze style navigation environments: Alien, Amidar, and Tutankham. For the hierarchical models, we set η to the same value at each level.

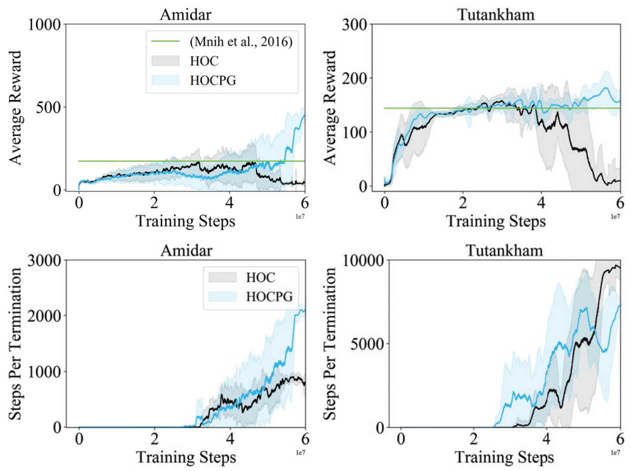


Figure 4: Hierarchical option-critic style model performance and steps per termination of the highest level option during learning with a η schedule.

In Figure 3, we report results for OCPG and OC across Alien, Amidar, and Tutankham. We include A3C performance from Table S3 of (Mnih et al. 2016) as a point of reference for how agents with primitive actions perform on these games. We report the results of a similar agent with a CNN and LSTM based representation like ours, but we should note this results are not directly comparable in a few ways. For example, the agents in past work were trained for considerably more steps. In early training, OCPG and OC have pretty much identical performance. This is particularly noticeable when $\eta = 0$ during the first 15 million steps. However, as η gradually increases, we begin to see OCPG significantly differentiate itself from OC. OC experiences significant instability once $\eta = 1.0$ at 45 million steps of training in all three games. Meanwhile, OCPG does not really see this instability or at least experiences it much less. Below our average reward results, we also provide insight about the termination behavior of the agents by plotting the average number of steps per termination over time. We can see that the gap between OCPG and OC consistently begins for each game when we start to see terminations become less frequent. Inline with our remarks in the last section, it is predictable that both algorithms should have the same performance when all options terminate every step. Additionally, it makes perfect sense that the gap between algorithms grows as η becomes high and options start taking a significant amount of time. When options are longer, OCPG has a superior ability to focus the policy over options on likely transition regions, allowing it to quickly adapt to and manage longer options.

We also report results for hierarchical option models in Figure 4 when using $N = 3$ levels of abstraction (standard option-critic uses $N = 2$). Unfortunately, both hierarchical models struggled to achieve the performance of a primitive

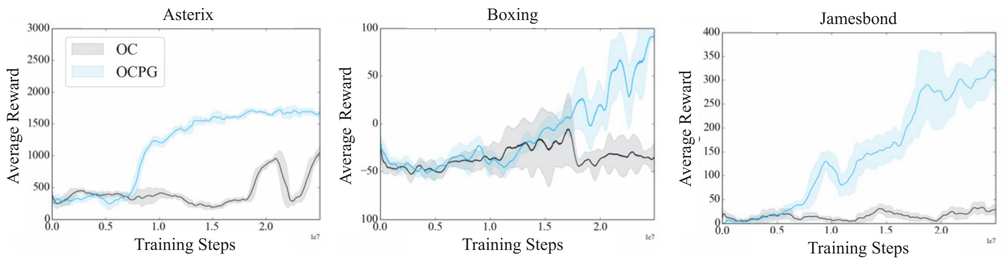


Figure 5: Option-critic style model performance for three Atari games with fixed $\eta = 0.3$.

action agent on Alien (not shown) and didn't seem to provide additional value over option-critic on the other games by using the third level of abstraction either (at least with only 60 million steps of training from scratch). Nonetheless, the model still surpasses prior results with primitive actions in a short amount of training and can still be useful for understanding the optimization behavior of the hierarchical option-critic architecture. Similar to the $N = 2$ case, we see significant instability in the HOC model that we do not see for HOCPG. Once again, in line with what we would expect, we see that the gap between the models only starts to grow when options start learning not to terminate. A noticeable difference with our option-critic experiments is that options become longer earlier. However, this could be related to our setting as η grows in effect when applied to more levels of abstraction, making the effective regularization higher at the same η.

Fixed Regularization

We would also like to verify that our proposed algorithms add value in the typical setting where η is simply set to some reasonable value. Based on our initial experiments we found that $\eta = 0.3$ was a good choice that led to options of non-trivial length for each game we explored. In addition to our three games from the previous section, we will explore four other Atari games that have quite different dynamics than these maze style navigation games including Asterix, Boxing, Jamesbond, and Tennis. We consider a sample efficient setting similar to (Pritzel et al. 2017) where we evaluate model performance after 10 million and 25 million training steps. This allows us to highlight the improved learning efficiency of OCPG when options are of non-trivial lengths.

In Table 1 we report results for OCPG and OC across 10 runs. While, again, results may not be directly comparable due to differences in the implementation, past work has reported performance for primitive action A3C after 10 million steps as 415.5 for Alien, 96.3 for Amidar, 301.6 for Asterix, 2.5 for Boxing, 31.5 for Jamesbond, -23.8 for Tennis, and 108.3 for Tutankham (Pritzel et al. 2017). In all but one case, OCPG results in superior performance to both OC and prior reported A3C results. The only apparent failure is Boxing at 10 million steps. That said, the performance achieved for Boxing at 25 million steps is very impressive given the amount of training and even surpasses some past asymptotic results reported for A3C (Mnih et al. 2016). Once again, we find that the strong comparative performance of OCPG is in a setting where options often do not terminate. For example, even by

Game	OC 10M / 25M	OCPG 10M / 25M
Alien	491.5 $\pm$ 115.3 / 618.8 $\pm$ 61.8	**663.0** $\pm$ 45.3 / **791.4** $\pm$ 44.7
Amidar	45.1 $\pm$ 26.4 / 89.4 $\pm$ 15.9	**219.3** $\pm$ 18.4 / **141.9** $\pm$ 18.6
Asterix	409.7 $\pm$ 76.6 / 1042.0 $\pm$ 98.1	**1213.4** $\pm$ 74.3 / **1679.8** $\pm$ 71.0
Boxing	-37.7 $\pm$ 14.7 / 32.9 $\pm$ 14.3	-41.4 $\pm$ 12.9 / **90.8** $\pm$ 6.8
Jamesbond	7.1 $\pm$ 5.5 / 29.3 $\pm$ 10.7	**118.6** $\pm$ 29.4 / **317.7** $\pm$ 38.7
Tennis	-17.9 $\pm$ 2.6 / -19.9 $\pm$ 1.3	**-14.4** $\pm$ 1.7 / **-13.1** $\pm$ 2.8
Tutankham	1.6 $\pm$ 2.5 / 0.8 $\pm$ 3.0	**16.8** $\pm$ 20.8 / **128.0** $\pm$ 27.3

Table 1: Average and standard deviation of the evaluation reward for Atari games across 10 runs, reported after 10 million steps (10M) and 25 million steps (25M). Bold indicates a statistically significant difference according to Welch's t-test.

10 million training steps, the average number of terminations per step for OCPG is 7.9 for Alien, 12.8 for Amidar, 8.3 for Asterix, 32.4 for Boxing, 5.5 for Jamesbond, 242.2 for Tennis, and 1266.3 for Tutankham.

In Figure 5, we highlight the learning behavior of OCPG and OC on three of the new games. For Asterix and Jamesbond, we see OCPG pretty immediately differentiate itself from the OC baseline and achieve quite impressive early performance. However, OCPG takes a little longer to differentiate itself from OC on Boxing. That being said, once it starts learning, it is able to achieve great performance quickly.

Discussion

Developing better general purpose methods for learning options from scratch is an important avenue of RL research. In this work, we reconsider an assumption in the intra-option policy gradient theorem and termination gradient theorem, namely that the policies and termination functions have independent parameters (Bacon, Harb, and Precup 2017). While this assumption is true in tabular settings, it is never the case for practical deep function approximation settings as parameter sharing across components of the model leads to sample efficient learning. We propose to rectify this issue by performing full architecture level *option-critic policy gradient* updates. A key distinction with prior approaches is that the update rule for the policy over options depends on the behavioral characteristics of the underlying options it selects over. We demonstrate that this modification leads to improved sample efficient learning across a test bed of Atari games. We show for a number of games that this update rule results in more stable and faster learning by focusing on a more representative state space distribution when options are long.

Acknowledgements

The authors thank Murray Campbell and Tim Klinger for fruitful discussions that helped shape this work. We also would like to thank Modjtaba Shokrian-Zini and Akshay Dharmavaram for their helpful comments and suggestions. Additionally, we thank Dan Jurafsky, Chris Potts, and Josh Greene for their support.

References

Bacon, P.-L.; Harb, J.; and Precup, D. 2017. The option-critic architecture. *AAAI*.

Bacon, P.-L. 2018. *Temporal Representation Learning*. Ph.D. Dissertation, McGill University Libraries.

Bellemare, M. G.; Naddaf, Y.; Veness, J.; and Bowling, M. 2013. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research* 47:253–279.

Caruana, R. 1997. Multitask learning. *Machine Learning* 28(1):41–75.

Colas, C.; Sigaud, O.; and Oudeyer, P.-Y. 2018. How many random seeds? statistical power analysis in deep reinforcement learning experiments. *arXiv preprint arXiv:1806.08295*.

Harb, J.; Bacon, P.-L.; Klissarov, M.; and Precup, D. 2017. When waiting is not an option: Learning options with a deliberation cost. *arXiv preprint arXiv:1709.04571*.

Klissarov, M.; Bacon, P.-L.; Harb, J.; and Precup, D. 2017. Learnings options end-to-end for continuous action tasks. *arXiv preprint arXiv:1712.00004*.

Kostas, J.; Nota, C.; and Thomas, P. S. 2019. Reinforcement learning without backpropagation or a clock. *arXiv preprint arXiv:1902.05650*.

Levy, K. Y., and Shimkin, N. 2011. Unified inter and intra options learning using policy gradient methods. In *European Workshop on Reinforcement Learning*, 153–164. Springer.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, 1928–1937.

Nachum, O.; Gu, S.; Lee, H.; and Levine, S. 2018. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*.

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

Pritzel, A.; Uria, B.; Srinivasan, S.; Badia, A. P.; Vinyals, O.; Hassabis, D.; Wierstra, D.; and Blundell, C. 2017. Neural episodic control. In *International Conference on Machine Learning*, 2827–2836.

Puterman, M. L. 1994. Markov decision processes: Discrete dynamic stochastic programming, 92–93.

Riemer, M.; Liu, M.; and Tesauro, G. 2018. Learning abstract options. *NIPS*.

Sutton, R. S.; McAllester, D. A.; Singh, S. P.; and Mansour, Y. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, 1057–1063.

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