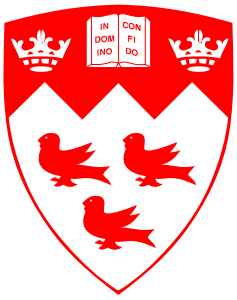
Diversity Enriched Option-Critic:

Learning Option Abstractions using Diversity in Option Behavior.

Anand Kamat



School of Computer Science

McGill University, Montreal

August 31, 2020

A thesis submitted to McGill University in partial fulfilment

of the requirements of the degree of Master of Science.

©Anand Kamat; 2020

Acknowledgements

Firstly, I would like to express my sincere gratitude to Prof. Doina Precup for giving me the opportunity to work under her supervision and guidance. I am deeply grateful for her precious insights and continuous support throughout my master's program. I applaud her enthusiasm to share her knowledge and expertise with the research community and her students. I would like to thank everyone in Mila and the Reasoning and Learning (RL) Lab with whom I had the privilege to work and share my enthusiasm; those who gave me the freedom to fail, think and grow to become a better version of myself. Last but not the least, I would like to thank my family: my parents and my sister for supporting me throughout my journey at McGill. Their confidence and belief in me has helped me every step of the way.

Abstract

Temporal abstraction allows reinforcement learning agents to represent knowledge and develop strategies over different temporal scales. The option-critic framework has been demonstrated to learn temporally extended actions, represented as options, end-to-end in a model-free setting. However, feasibility of option-critic remains limited due to two major challenges, multiple options adopting very similar behavior, or a shrinking set of options that are relevant to the task. These occurrences not only void the need for temporal abstraction, they also suppress performance. This thesis proposes an approach to tackle these problems by learning a *diverse set of options* online. We introduce an information-theoretic intrinsic reward, which augments the task reward, as well as a novel termination objective, in order to encourage diversity in the option set. We show empirically that our proposed approach achieves state-of-the-art performance for learning options on several discrete and continuous control tasks, not only outperforming option-critic by a wide margin, but also PPO. Furthermore , we show that our approach sustainably generates robust, reusable, reliable and interpretable options, in contrast to option-critic.

Résumé

Les actions abstraites temporellement aide agents d'apprentissage par renforcement représenter les connaissances et élaborer des stratégies sur une large gamme d'échelle temporelle. Il a été démontré que le cadre d'option-critic apprend des actions étendues dans le temps, représentées comme des options, de bout en bout dans un environnement modèle-free. Cependant, la faisabilité de option-critic reste limitée en raison de deux défis majeurs, de multiples options adoptant un comportement très similaire ou d'un ensemble réduit d'options pertinentes pour la tâche. Ces événements annulent non seulement le besoin d'abstraction temporelle, mais suppriment également les performances. Cette thèse propose une approche pour aborder ces problèmes en apprendre un ensemble diversifié d'options. Nous introduisons une récompense intrinsèque l'information-théorique, qui augmente la récompense de la tâche, ainsi qu'un nouvel objectif de résiliation, afin d'encourager la diversité dans l'ensemble d'options. Nous montrons empiriquement que notre approche proposée permet d'atteindre des performances de pointe sur plusieurs tâches de contrôle discrètes et continues, non seulement surpassant de loin l'option-critic, mais également PPO. De plus, nous montrons que notre approche génère durablement des options robustes, réutilisables, fiables et interprétables, contrairement à l'option-critic.

Contents

List of Tables

List of Figures

1

Introduction

1.1 Motivation

Humans are capable of learning very complex real world tasks owing to our ability to conceive actions over a wide range of temporal scales. Consider a task of going to work in the morning. This task can be decomposed into sub tasks such as: getting ready, having breakfast and driving to work. Each of these lower level actions may require varying time steps to complete. Now, we can further decompose getting ready into tasks such as brushing the teeth, taking a shower, getting dressed and so on. Similarly, it is possible to keep breaking down these tasks to even simpler ones up until the exact motion of limbs or even the contraction of muscle fibres required to perform them. This example illustrates how humans carry out everyday tasks by thinking hierarchically and creating objectives over a different temporal scales. Making decisions at each step requires autonomous foresight and planning over multiple time scales. The ability to think and plan hierarchically helps us explicate and solve problems by reducing complexity of choosing actions while generating shorter and more interpretable plans. It is interesting to note that humans can learn and plan tasks over several temporal scales without any explicit instructions on how to create sub tasks and strategize over them. Learning and planning at multiple levels of temporal abstraction has a strong appeal to the reinforcement learning and artificial

intelligence research community.

There has been a lot of interest in hierarchical reinforcement learning and temporal abstraction for the past few decades (Sacerdoti 1974; Sutton 1985; Iba 1989; Korf 1983). Temporal abstraction has shown to improve exploration (Mann, Mankowitz, and Mannor 2014) in learning, benefit planning, increase robustness to environment perturbations and aid in transfer problems (Sutton, Precup, and Singh 1999; Bacon, Harb, and Precup 2017). Decomposing task over multiple temporal scales also improves interpretibility (Sutton, Precup, and Singh 1999). Over the years several approaches to conceptualize temporal abstraction for reinforcement learning have been proposed (Singh 1992; Parr and Russell 1998; Sutton, Precup, and Singh 1999; Vezhnevets et al. 2017). The option framework (Sutton, Precup, and Singh 1999; Precup 2000) is one of the most effective formalization of temporal abstraction to learn and plan with temporally extended actions. An option is a subroutine which is selected to run until a specified condition is met, upon which it terminates and allows a chance for a better option to be selected.

The option-critic architecture (Bacon, Harb, and Precup 2017) is a gradient-based framework capable of learning both the intra-option policies as well as the termination functions end-to-end without explicitly providing any sub-goals. Option-critic demonstrated the effectiveness of temporal abstraction in learning interpretable and reusable skills on several tasks. However, as with most reinforcement learning algorithms, option-critic solely relies on rewards explicitly received from the environment based on the agent's experience. As a consequence, options often begin collapsing in various ways, for example options becoming primitive actions, a single option learning to solve the entire task and dominating the others, or several options becoming similar. These degeneracies negatively impact the agent's ability to re-use the learned option set in new situations. Such degeneration is to be expected as all Markov-Decision Processes (MDPs) can be solved using primitive actions. However, degeneracies void the need for temporal abstraction and heirarchial learning in reinforcement learn-

ing agents. There have been attempts to tackle the problem of options collapsing onto multiple copies of the optimal policy (Bacon, Harb, and Precup 2017; Harutyunyan et al. 2019) as well as ensuring options do not shrink too much over time (Harb et al. 2018). However, finding a solution that can easily generalize over a wide range of tasks is still an ongoing challenge. Furthermore, for most of the techniques used for learning options, the agent lacks motivation for learning useful, robust yet interpretable options since maximizing expected returns is all that drives the learning framework. The need for temporal abstraction is best realized when the agent learns specialized options intuitively which can be effectively reused and exploited for learning a task.

In this thesis, we tackle the problem of constructing a *diverse set of options*. *Diversity* in options refers to options learning mutually distinct behavior. An option's behavior is indicated by its intra-option policy. Diverse options can be exploited to increase exploration as well as robustness in learning challenging tasks (Gregor, Rezende, and Wierstra 2016; Eysenbach et al. 2018; Harutyunyan et al. 2019). A common approach for encouraging diversity in a policy is entropy regularization (Williams and Peng 1991; Mnih et al. 2016), but it does not capture the idea of the set of options itself containing skills that differ from each other and hence often fails to produce desired results. We use intrinsic motivation to address this issue. Using auxiliary rewards has shown encouraging results promoting useful properties such as exploration and performance (Ng, Harada, and Russell 1999; Singh et al. 2010; Bellemare et al. 2016). In this work, we introduce an auxiliary reward which, when combined with the task reward encourages diversity in the policies of options. We empirically show how this diversity can help options explore better on several standard continuous control tasks.

We then focus on the termination component of options. Option critic's termination objective (Bacon, Harb, and Precup 2017) aims to maximize expected returns by validating the option with the highest value at a given state. Unfortunately, this prevents the sub-optimal option to be selected and trained which is often the cause for

option degeneration. Our approach suggests that instead of having options compete for selection, adequate exploration of all available options should be encouraged so long as they exhibit diverse behavior. In this work, we propose a novel information theoretic termination objective which, coupled with our proposed intrinsic reward bonus , produces robust, task relevant options. We test this new objective quantitatively and qualitatively in a classic tabular setting as well as several standard discrete and continuous control tasks.

1.2 Thesis Overview

The thesis begins with motivation behind pursuing this which is articulated in this chapter. The required prerequisites are covered in Chapter 2. The background begins with a basic overview of reinforcement learning and various techniques developed in the field such as Policy Iteration, Temporal Difference Learning, On-Policy and Off-Policy Methods and Policy Gradient. Section 2.2 outlines temporal abstraction and the options framework in the context of learning temporally extended actions in reinforcement learning. The option-critic architecture is presented as an end-to-end learning framework capable of learning the components of options such as intra-option policies as well as the termination conditions without explicitly defined sub-goals. Section 2.3 focuses on intrinsic motivation and its benefits in model-free reinforcement learning algorithms. The section also reiterates relevant literature in the field that support its validity in influencing the way agent behave. Concepts of information theory are described in section 2.4. Section 2.5 contains a brief overview of Deep Reinforcement Learning (Deep RL). The section highlights recent developments in deep RL and presents algorithms such as Proximal Policy Optimization (PPO) which are used in the empirical analysis for this thesis.

The thesis revolves around the idea of diversity in options as a significant factor in learning useful, robust and interpretibile options. Chapter 3 explores how diversity

directly affects the way options are learned in a model-free setting. Using concepts of information theory, a pseudo-reward is derived which is coupled with the task reward to encourage overall diversity in options. The benefits and effects of incorporating diversity directly into the reward signal are studied using simulated experiments on continuous control tasks.

Chapter 4 focuses on diversity to define the scope of options. A termination objective built around the idea of identifying states where options can grow most diverse is proposed. The new termination objective coupled with the proposed intrinsic motivator not only encourage diversity in options, but also drastically improves overall performance. The termination objective also offers additional benefits such as promoting options to remain relevant to the task, better transfer capabilities, and improved interpretability of option's behavior. All the above mentioned properties are empirically tested on a wide range of tasks to show a consistent and general learning pattern.

Finally, chapter 5 presents the highlights of the thesis. The conclusion also mentions the shortcomings as well as viable future direction for this work.

2

Background

2.1 Reinforcement Learning

Reinforcement learning (RL) is a framework inspired by behavioral psychology (Sutton 1985) developed to learn tasks involving sequential decision making. It involves taking sequential actions which are learned from experience gathered by interacting in an environment. Much like trial-and-error learning, an agent attempts various actions on the environment and strengthens actions yielding positive or successful responses, given in the form of rewards, received from the environment. The agent's objective is to maximize the cumulative rewards.

The agent-environment interaction process is visualized in Fig 2.1 (Sutton and Barto 2018).

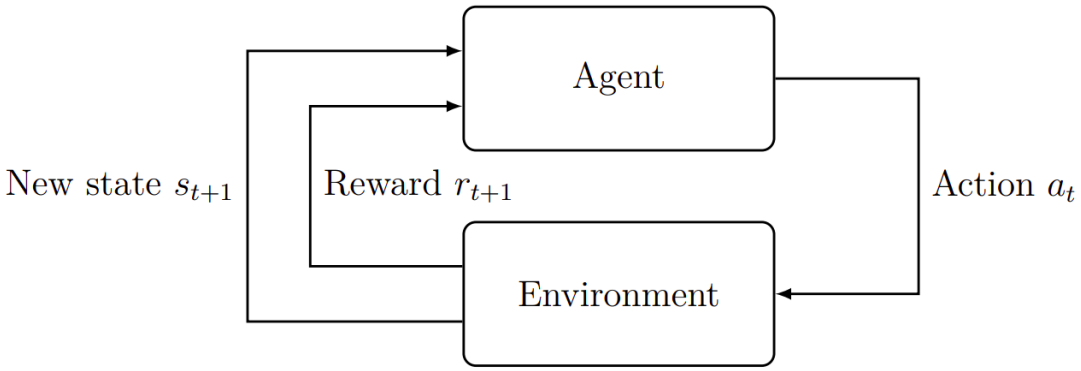


Figure 2.1: Agent-environment Interaction in RL

The problem setting in reinforcement learning is modeled as a Markov Decision Process (MDP) which consists of:

- Set of states $\mathcal{S}$
- Set of actions $\mathcal{A}$
- Scalar discount factor $\gamma \in [0, 1)$
- Reward function r : $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$
- A transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,\ 1]$

Policy (π) is a function which defines how an agent selects actions. A stochastic policy is defined as $\pi : \mathcal{S} \times \mathcal{A} \to [0,\ 1]$ i.e the policy outputs the probability of taking an action A_t at state S_t. In this thesis we focus on the objective of learning a policy to optimize the expected return $V_\pi(s)$: $\mathcal{S} \to \mathbb{R}$ which is called the *value function*.

$$\begin{aligned} V_\pi(s) &= \mathbb{E}_\pi \left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + ... \right] \\ &= \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| S_t = s \right] \\ &= \mathbb{E}_\pi \left[r_{t+1} + \gamma V_\pi(s_{t+1}) \middle| S_t = s \right] \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s_{t+1} = s', r|s, a) \left[r + \gamma V_\pi(s') \right], \forall s \in \mathcal{S} \end{aligned} \tag{2.1}$$

where $\mathbb{E}_\pi[\cdot]$ denotes the expected value of the term. It is possible to bootstrap the value estimate of the next state to evaluate the current value function. This recursive relationship create a Bellman equation (Bellman 1954) which is often exploited in reinforcement learning. The bellman equation can be solved to obtain optimal value

function shown in Eq 2.2. The optimal value function V*(s) can be defined as the cumulative discounted return following an optimal policy.

$$\begin{aligned} V^*(s) &= \max_{\pi} V_\pi(s) \\ &= \max_{a} \mathbb{E}\left[r_{t+1} + \gamma V^*(s_{t+1}) \middle| S_t = s, A_t = a\right] \\ &= \max_{a} p(s_{t+1} = s', r|s, a)\left[r + \gamma V^*(s')\right] \end{aligned} \tag{2.2}$$

Since the above value function ($V_\pi(s)$) depends on the state, it is also referred to as state-value function. We can also define the state-action value function ($Q_\pi(s, a)$) also referred to as Q-value function.

$$\begin{aligned} Q_\pi(s, a) &= \mathbb{E}_\pi\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + ...\right] \\ &= \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| S_t = s, A_t = a\right] \\ &= r(s, a) + \gamma \sum_{s'} p(s_{t+1} = s'|s_t = s, a_t = a) \sum_{a'} \pi(s_{t+1}, a') Q_\pi(s', a') \end{aligned} \tag{2.3}$$

$$\begin{aligned} Q^*(s, a) &= \max_{\pi} Q_\pi(s, a) \\ &= r(s, a) + \gamma \sum_{s'} p(s_{t+1} = s'|s_t = s, a_t = a) \max_{a'} Q(s', a') \end{aligned}$$

The optimal Q-value (Q*(s,a) is the expected discounted return following the optimal policy (π^*) given a state s and an action a. It is often useful to use the advantage function $A_\pi(s, a)$ which measures how useful action a is compared to the expected return when following a policy π.

$$A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s) \tag{2.4}$$

2.1.1 Policy Iteration

The key idea of reinforcement learning is to exploit value functions to search and learn good policies. In general, this process is broken down into two phases: the

policy evaluation phase often referred to as the prediction problem in RL, and the policy improvement phase or the control problem in RL. Policy evaluation refers to computing the value function for a policy π. The value function assesses the policy and is useful in finding better policies. In the case of policy improvement, the value function is used to greedily update the policy π such that the updated policy π', is better than the former policy ($V_{\pi'}(s) \geq V_{\pi}(s)$). The policy iteration involves two simultaneous processes, the policy evaluation and the policy iteration. The interacting processes are shown in Figure 2.2 (Sutton and Barto 2018). The objective of RL is to learn a policy so as to maximize the expected return.

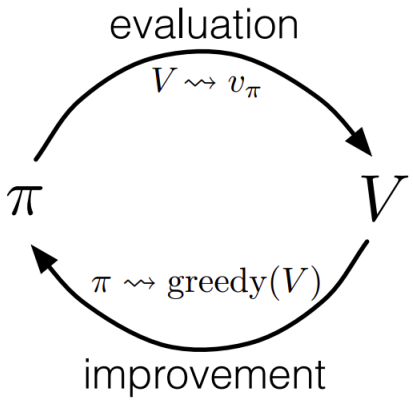


Figure 2.2: Policy Iteration

Policy iteration without explicit knowledge of the environment dynamics or a model is referred to as model-free RL. If the environment transitions and the reward function are known, the algorithm is referred to as model-based RL. Learning in RL can occur in an online as well as an offline case. Offline learning is used when environment data is limited where as in the case of online learning the agent is able to interact with the environment to gather data. In this thesis, we focus on the online learning problem. A challenge faced in online learning is sample efficiency, which is to learn a generalized policy without requiring large amount of data.

2.1.2 Temporal Difference Learning

Temporal difference (TD) learning is a central concept in RL. TD learning combines ideas from Monte Carlo, which involves learning from gathering experience, and dynamic programming, involving bootstrapping estimates from other estimates without waiting for the final outcome. The TD prediction update (Sutton and Barto 2018) is given as:

$$V(S_t) = V(S_t) + \alpha \left[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right] \tag{2.5}$$

where α is a learning rate. This update is called the TD(0) or *one-step* update as only the value-estimate from the next state is used in each update. The term $R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ is called the TD error. The TD(λ) algorithm averages multi-step or n-step updates. The n-step updates are each weighted proportional to $lambda^{n-1}$, where $\lambda \in$ [0,1] and is normalized by a factor of 1-λ. λ-return (Sutton and Barto 2018) is defined as:

$$G_t^{\lambda} = (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_{t:t+n} \tag{2.6}$$

2.1.3 On-Policy and Off-Policy Methods

There are two common approaches used in control tasks: on-policy methods and off-policy methods. On-policy methods involve evaluating or improving the policy that is used to interact with the environment, whereas off-policy methods evaluate or improve a policy different from that used to generate the data (Sutton and Barto 2018). In this thesis, we use on-policy methods for all our algorithms.

A common on-policy TD control method is Sarsa (Sutton and Barto 2018). Instead of the state value function $V_\pi(s)$ for the current behavior policy, sarsa estimates the state value function $Q_\pi(s, a)$. The one step (TD(0)) update is given by:

$$Q_\pi(s_t, a_t) \rightarrow Q_\pi(s_t, a_t) + \alpha \left[r_{t+1} + \gamma Q_\pi(s_{t+1}, a_{t+1}) - Q_\pi(s_t, a_t)\right] \tag{2.7}$$

The update is done after every transition from a non-terminal state (S_t). If (S_{t+1}) is terminal, then $Q_\pi(s_{t+1}, a_{t+1})$ is defined as zero. This method gets its name from the quintuple of events $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$ which spells out *Sarsa*.

While on-policy methods learn the same policy which interacts with the environment, off-policy methods attempt to estimate and improve a target policy while a separate policy called behavior policy, is used to generate the experience data by interacting with the environment. This behavior policy is often an exploratory policy. Off-policy methods use a correction term to account for the discrepancy between the two policies. One of the most common techniques is to use importance sampling. The importance sampling ratio for a trajectory starting from time t to T is given by:

$$\rho_{t:T-1} = \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{\mu(A_k|S_k)} \tag{2.8}$$

where π is the target policy and μ is the behavior policy. The importance sampling ratio helps estimate the expected value of the target policy using returns from a behavior policy:

$$\mathbb{E}\left[\rho_{t:T-1} G_t | S_t\right] = V_\pi(S_t) \tag{2.9}$$

One of the most popular off-policy TD control method is the Q-Learning method (Watkins and Dayan 1989). The one-step update is given as:

$$Q_\pi(s_t, a_t) \rightarrow Q_\pi(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q_\pi(s_{t+1}, a) - Q_\pi(s_t, a_t) \right] \tag{2.10}$$

The action value function Q, directly approximates the optimal action-value function Q* independent of the policy being followed (Sutton and Barto 2018). Q-Learning is considered off-policy as the state value function Q learns from actions that may be outside the scope the target policy. In this thesis on-policy methods are used for empirical evaluations both in the tabular as well as the non-linear function approximation case.

2.1.4 Policy Gradient

So far we have considered tabular methods where it is possible to calculate and store the values for each state of the MDP. Tabular cases use a lookup table to store Q(s,a) values for each state action pair. However, these cases don't easily scale to tasks with arbitrarily large and high dimensional state space. An example of such tasks are navigation tasks with image data as input. We cannot expect to find an optimal policy or the optimal value function even with a limit of infinite computational time. Instead, it is useful to learn a function which approximates the solution using limited computational resources.

Policy gradient methods learn a parameterized policy that can act on an environment without consulting a value function. Although a value function may still be used to learn a policy, it is not required to select an action. A parameterized policy is represented by $\pi(a|s,\theta) \in [0,1]$ for the probability that action a is taken given the current state s with parameter $\theta \in \mathbb{R}^{d'}$. The policy parameters are updated based on the gradient of a performance objective $\mathcal{J}(\theta)$ with respect to the policy parameter.

$$\theta_{t+1} = \theta_t + \alpha \widehat{\nabla \mathcal{J}(\theta_t)} \tag{2.11}$$

where $\widehat{\nabla \mathcal{J}(\theta_t)}$ is the stochastic estimate whose gradient approximates the gradient of the performance measure (Sutton and Barto 2018). Policy gradient methods apply as long as $\pi(a|s,\theta)$ is differentiable with respect to its parameters i.e. as long as $\nabla_\theta \pi(a|s,\theta)$ exists and is always finite. Actor-critic methods learn both a policy and a value function. Actor-critic methods are online, incremental learning algorithms.

The one step actor critic update (Sutton and Barto 2018) is:

$$\begin{aligned}\theta_{t+1} &= \theta_t + \alpha \left(G_t - \hat{V}(S_t, w)\right) \frac{\nabla_\theta \pi(A_t|S_t, \theta_t)}{\pi(A_t|S_t, \theta_t)} \\ &= \theta_t + \alpha \left(R_{t+1} + \gamma \hat{V}(S_{t+1}, w) - \hat{V}(S_t, w)\right) \frac{\nabla_\theta \pi(A_t|S_t, \theta_t)}{\pi(A_t|S_t, \theta_t)} \\ &= \theta_t + \alpha \delta_t \frac{\nabla_\theta \pi(A_t|S_t, \theta_t)}{\pi(A_t|S_t, \theta_t)}\end{aligned} \tag{2.12}$$

2.2 Temporal Abstraction

Humans have a remarkable ability to overcome arduous tasks by learning, planning and representing knowledge hierarchically. Temporal abstraction can help scale reinforcement learning tasks by decomposing the problem space over different temporal scales. There has been a lot of development in formalizing temporal abstraction for reinforcement learning. The MAXQ method (Dietterich 1998) decomposes the target MDP into a hierarchy of smaller MDPs. The value function of the target MDP was computed using a combination of the value functions of the smaller MDPs. Hierarchical abstract machines (HAM) (Parr and Russell 1998) learns hierarchies by imposing constraints on the policies. Developing abstract MDPs (Hauskrecht et al. 1998) has shown to break the original MDP's state space allowing macro actions to learn policies over specific regions. The abstract MDP approximates the original MDP in such a way that it can be solved more efficiently. FeUdal networks (Vezhnevets et al. 2017) proposed an architecture to learn policies belonging to a hierarchy. The framework employs a manager module and the worker module. The manager generates goals which are enacted by the worker. FeUdal networks were shown to be advantageous in long-term credit assignment tasks. Hierarchical Actor-Critic (HAC) (Levy, Platt, and Saenko 2018) framework was shown to effectively learn multiple levels of hierarchy simultaneously accelerating learning when compared to non-hierarchical frameworks. The options framework (Sutton, Precup, and Singh 1999; Precup 2000) is one of the most widely used frameworks formalizing temporal abstraction. Options are tempo-

rally extended actions which act on a lower temporal resolution. Options and an MDP together form a semi-Markov Decision Process (SMDP) (Sutton, Precup, and Singh 1999). Hence the option framework inherit most of the useful properties of SMDPs for learning variable length macro-actions or options.

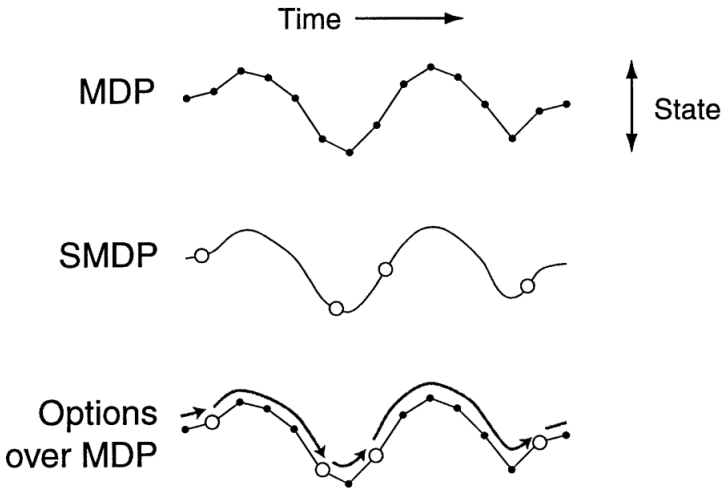


Figure 2.3: Options help analyze the underlying MDPs which constitute the SMDP (Sutton, Precup, and Singh 1999). SMDPs emulate temporal abstraction as the can be composed by variable length transitions.

Option is a triple $(\mathcal{I}_o, \pi_o, \beta_o)$ defined using:

- Initiation set $\mathcal{I}_o \subseteq \mathcal{S}$
- Intra-option policy $\pi_o : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$
- Termination function $\beta_o : \mathcal{S} \rightarrow [0, 1]$

An option o can be selected at a state permitted by the initiation set, i.e s if $s \in \mathcal{S}$ and $s \in \mathcal{I}_o$. Upon selection, the agent's behavior under an option is depended on the option's policy called the *Intra-option* policy. The termination function β_o determines if the option terminates, upon which the agent selects another option. This *call-and-return* model executes until the episode terminates. The policy which selects an option from the set of options is referred to as policy over options (π_Ω). Primitive action frameworks can be seen to be a special form of the options framework where

each option terminates in one step. In this thesis, we assume that all options are available everywhere such that $\mathcal{I}_{o \in \mathcal{O}} = \mathcal{S}$ where $\mathcal{O}$ is the set of all options.

Learning options end-to-end has been a big challenge mainly because it is difficult to define what constitutes a good option. Several approaches have been proposed to tackle the option discovery problem (Mann, Mankowitz, and Mannor 2014; Stolle and Precup 2002). The option-critic architecture is a gradient-based framework capable of learning options end-to-end. The intra-option policies as well as the termination functions are learned using the expected return objective. All the options are learned simultaneously in an online, model-free setting. The intra-option policy weights and the termination weights are assumed to be independent. Option-critic empirically demonstrated the benefits of temporal abstractions in context of exploration, reusability of specialized skills and interpretability of options.

The option-value function $Q_\Omega(s, o) = \mathcal{S} \times \mathcal{O} \rightarrow \mathbb{R}$ is the value of executing an option in the context of state and option (Bacon, Harb, and Precup 2017)

$$Q_\Omega(s, o) = \sum_a \pi_{o,a}(a|s) Q_U(s, o, a) \tag{2.13}$$

where $Q_U : \mathcal{S} \times \mathcal{O} \times \mathcal{A} \rightarrow \mathbb{R}$ is the value of executing an action in the context of a state-option pair.

$$Q_U(s, o, a) = r(s, a) + \gamma \sum_{s'} p(s'|s, a) U(o, s') \tag{2.14}$$

The function $U(o, s') : \mathcal{O} \times \mathcal{S} \rightarrow \mathbb{R}$ is the option value upon entering state s' (Sutton, Precup, and Singh 1999).

$$U(o, s') = (1 - \beta_o(s')) Q_\Omega(s', o) + \beta_o V_\Omega(s') \tag{2.15}$$

Option critic derived the intra-option gradient optimizing the expected return objective. The intra-option policy gradient theorum (Bacon, Harb, and Precup 2017) states that given a set of Markov options with stochastic and differentiable intra-

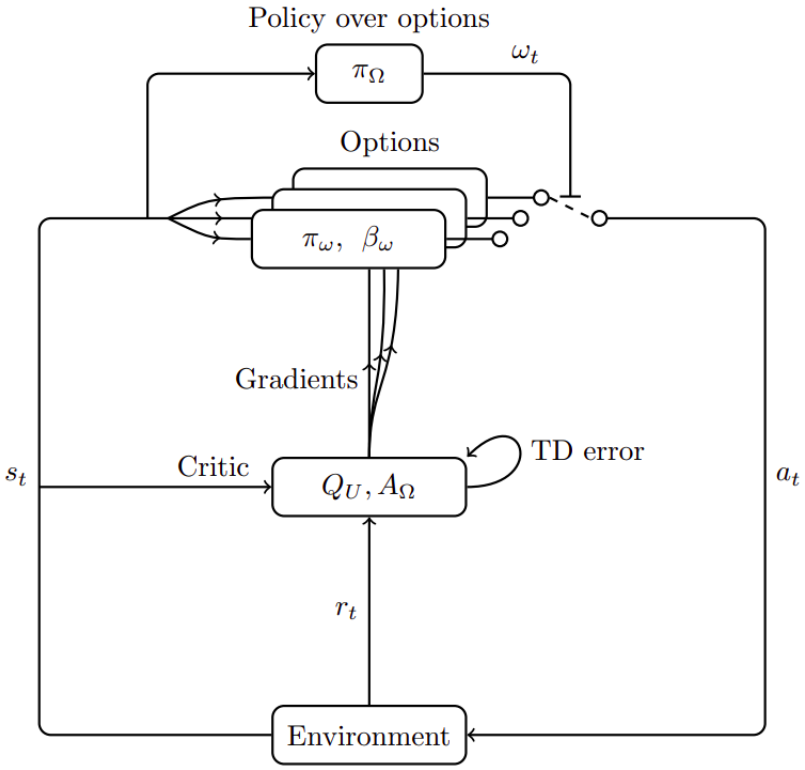


Figure 2.4: **Option-Critic Architecture** (Bacon, Harb, and Precup 2017). The switch indicating the active option ω is selected by the policy over option. Only when the option terminates does the policy over option select a new option.

option policies, the gradient of the expected discounted return with respect to their parameters θ and initial condition (s_0, o_0) is:

$$\sum_{s,o} \mu_\Omega(s, o|s_0, o_0) \sum_a \frac{\partial \pi_{o,\theta}(a|s)}{\partial \theta} Q_U(s, o, a) \tag{2.16}$$

where $\mu_\Omega(s, o|s_0, o_0)$ is a discounted weighting of state-option pairs starting from (s_0, o_0).

The termination gradient theorem (Bacon, Harb, and Precup 2017) states that given a set of Markov options with stochastic and differentiable termination functions, the gradient of the expected discounted return with respect to their parameters ν and

initial condition (s_1, o_0) is:

$$-\sum_{s',o} \mu_\Omega(s', o|s_1, o_0) \frac{\partial \beta_{o,\nu}(s')}{\partial \nu} A_\Omega(s', o) \tag{2.17}$$

where $A_\Omega(s', o)$ is the advantage function over options. $A_\Omega(s', o) = Q(s', o) - V_\Omega(s')$. The advantage function offers and intuitive interpretation to the gradient update. If the value of the executing option is sub-optimal with respect to the expected value of all options, the gradient correction increases thereby promoting termination. The termination update hence validates the best possible option at each state. The paper illustrates in a simple tabular task how option terminations aid interpretable and specialized abstractions which can be reused in similar tasks.

However, since theoretically any MDPs can be solved using primitive actions, options began defaulting to primitive actions. This phenomenon is commonly referred to as degeneracy in options. Options may degenerate in two ways: a single option performs the entire task without ever terminating, or options may terminate in every step and keep switching at every step. Degeneracy is counterproductive as it voids the need for temporal abstraction in any task along with surrendering all the useful properties of learning options. In an effort to prolong degeneration, a deliberation cost inspired by bounded rantionality framework (Harb et al. 2018), was introduced which penalized frequent option terminations. Although deliberation cost succeeded in delaying degeneracy, it was heavily depended on the parameter.

Another limitation of termination gradient can be attributed to the presence of the advantage function in the update, which validates the option with the highest value without exploring the other options sufficiently. Selecting the best available option without adequately exploring others isn't a good strategy. Termination critic (Harutyunyan et al. 2019) emphasised the need to decouple the terminations from the expected return objective. The paper proposed that terminations should focus on the compressibility of the option's encoding.

The termination gradient also doesn't prioritize learning the variety of states where

options need to focus their terminations. Identifying and exploiting *critical* states where options can develop better and useful abstractions can improve exploration, performance, transfer abilities as well as interpretability.

In this thesis, we draw attention to *diversity in options*. Diversity in options refers to options learning mutually distinct behavior as indicated by its intra-option policy. Current practices involve using an entropy regularizer (Williams and Peng 1991; Mnih et al. 2016) for policy over options updates which increases the stochasticity of the policy encouraging all options to be explored sufficiently. However, since entropy regularizer doesn't relate to the intra-option policies directly, it often fails to produce the desired results. We propose that diversity in options can be used for defining the scope of option's abstractions. Our method identifies certain critical states in the trajectory which are capable of generating diverse option trajectories and encourages options to terminate there. These states represent regions where the agent can exploit diversity in options for exploration and stability. Over the years, identifying bottleneck states as useful sub-goals for directing options, has had lot of success (McGovern and Barto 2001; Stolle and Precup 2002; Bacon 2013). Our approach can be seen to target the bottleneck states characterized by diversity in options. Unsupervised skill discovery has shown to be capable of learning challenging tasks using an information theoretic objective (Eysenbach et al. 2018). Our algorithm however, is also capable of learning the task simultaneously. Combining deep skill chaining with option-framework autonomously was shown to generate sequentially connected and spatially localized options in a model-free setting (Bagaria and Konidaris 2020). In our approach, we do not impose any restriction on option's initiation which is quite useful in transfer settings.

2.3 Intrinsic Motivation

Reinforcement learning usually involves maximizing expected return based on the rewards explicitly received from the environment. However, in many instances the reward function fails to capture all the necessary information that may help the agent's decision. Use of conventional reinforcement learning algorithms is often limited to learning overly simplified tasks or tasks which require a large amount of training experience. More often than not, the agent is required to learn tasks where these properties are rarely observed. Large and complex task domains lead to poor sample efficiency and the agent may even fail to explore the environment exhaustively. Algorithms, in such instances, can benefit from complementary incentives drawn from intrinsic motivation. Intrinsic motivation has been researched extensively in psychology and neuroscience (Harackiewicz and Elliot 1993; Sansone and Harackiewicz 2000; Deci and Ryan 2010). Intrinsic motivation is defined as the incentive drawn from a source separate from the task incentives. Intrinsic motivation and reward modifications have been very successful in inducing certain desirable properties in reinforcement learning algorithms. A common one step update in RL using reward augmentation can be represented by the following formula:

$$Q_\pi(s_t, a_t) \leftarrow Q_\pi(s_t, a_t) + \alpha \left[r_{t+1} + b_{t+1} + \gamma Q_\pi(s_{t+1}, a_{t+1}) - Q_\pi(s_t, a_t) \right] \tag{2.18}$$

where r_t is the task reward and b_t is the pseudo reward. Reward shaping for the computational-oriented perspective of reinforcement learning can be seen as analogous to the concept of operation conditioning from psychology. Reward shaping can aid the temporal credit assignment problem by making correct behavior apparent through immediate and localised incentives.

Count based pseudo rewards have shown to improve exploration in several tasks (Bellemare et al. 2016; Ostrovski et al. 2017; Tang et al. 2017; Machado, Bellemare, and Bowling 2018). Pseudo-counts refer to approximate state visitation counts by the agent during its trajectories. This measure can be useful to encourage the agent

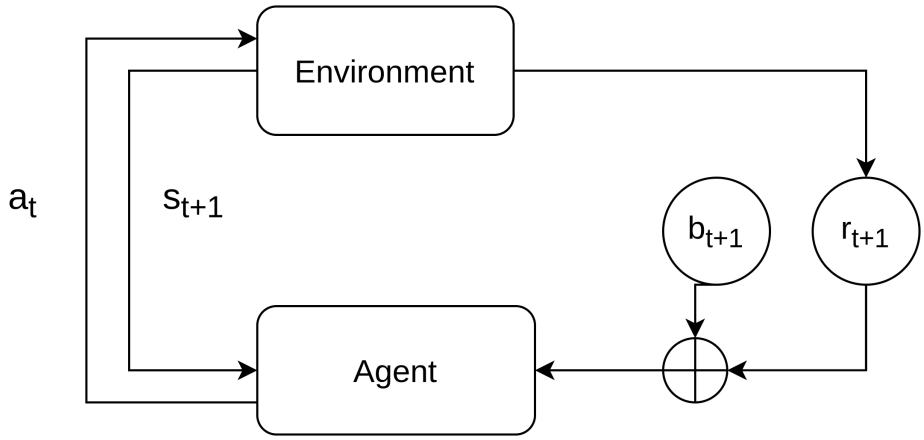


Figure 2.5: **Reinforcement Learning with reward shaping.** The agent seeks to maximize the returns as a combination of task reward r_t and pseudo-reward b_t. The pseudo-reward may be dependent on of independent from the environment based on how it is defined. However, the pseudo-reward doesn't directly relate to the task problem that the agent is attempting to solve.

to explore states which have been inadequately explored. Self-supervised prediction errors have also shown to help the agent explore better. Ng, Harada, and Russell 1999 formulated reward function modifications which leaves the optimal policy unaffected. Zheng, Oh, and Singh 2018 defined a intrinsic reward function whose parameters can be learned without providing any knowledge of the task. Coupling primary task reward with intrinsic motivation can also increase robustness and generalization (Singh et al. 2010). Intrinsic motivation can even replace task reward to engineer task independent behavior. Skills learned independent of the task reward can used as a prior to learn challenging tasks (Eysenbach et al. 2018). Our work on intrinsic motivation draws inspiration from similar methods to encourage diversity in options with the major difference of distinguishing options through their behaviors instead of states. In our approach we use a pseudo-reward bonus to encourage options to learn diverse behaviors.

2.4 Information Theory

Information theory is a field that primarily deals with deriving fundamental theoretical limits on achievable performance when communicating information over channels and then development of schemes that provides performance that is reasonable good in comparison with the optimal performance as given by the theory. Information theory can be viewed as a branch of applied probability theory. Information theory was introduced to find limits on signal processing and communications operations such as data compression (Shannon 1948). A key measure of information theory is *entropy*. Entropy measures a random variable's (RV) self-information or uncertainty inherent in the random variable. Self-information is the level of information associated with an event or the outcome of a random variable. Given a random variable X, with possible outcomes x_i, each with probability $P_X(x_i)$, the entropy $\mathcal{H}(X)$ of X is given by:

$$\begin{aligned}\mathcal{H}(X) &= -\sum_i P_X(x_i) log P_X(x_i) \\ &= \sum_i P_X(x_i) I_X(x_i) \\ &= \mathbb{E}[I_X] \\ &= \mathbb{E}\left[-log(P(X))\right]\end{aligned} \tag{2.19}$$

where I_X is the self-information of the random variable X and $I_X(x_i)$ is the self-information related to a particular outcome. For a stochastic source of data, a lower probability value of an event occurring carries more information than high probability events. Entropy hence relates to the information of a random variable. Information and entropy have also been used to measure the unpredictability of an outcome.

Joint entropy is the measure of uncertainty associated with a set of random variables. The joint entropy $\mathcal{H}$(X,Y) of two random variables X and Y is given as:

$$\mathcal{H}(X,Y) = -\sum_{x\in X}\sum_{y\in Y} P(x,y) log[P(x,y)] \tag{2.20}$$

The conditional entropy measures the information associated for the outcome of a random variable X given a random variable Y. The conditional entropy $\mathcal{H}$(X|Y) is given by:

$$\mathcal{H}(X|Y) = -\sum_{y\in Y} p(y) \sum_{x\in X} p(x|y) log p(x|y) = -\sum_{x,y} p(x,y) log p(x|y) \tag{2.21}$$

Mutual information measures the amount of information that can be obtained about one random variable by observing another. The mutual information $\mathcal{I}$ of X relative to Y is given by:

$$\mathcal{I}(X;Y) = \sum_{y\in Y} \sum_{x\in X} p(x,y) log \frac{p(x,y)}{p(x)p(y)} \tag{2.22}$$

A few useful properties in information theory are:

$$\mathcal{H}(X|Y) = \mathcal{H}(X,Y) - \mathcal{H}(Y) \tag{2.23}$$

$$\begin{aligned} \mathcal{I}(X;Y) &= \mathcal{H}(X) - \mathcal{H}(X|Y) \\ &= \mathcal{H}(X) + \mathcal{H}(Y) - \mathcal{H}(X,Y) \\ &= \mathcal{I}(Y;X) \end{aligned} \tag{2.24}$$

Cross entropy between two probability distributions p and q measures the average information for an estimated optimized probability distribution q rather than true distribution p. Cross entropy is commonly used to measure the difference between two probability distributions. Cross entropy can be seen to be related to Kullback-Leibler divergence or KL divergence. KL divergence between two distributions P and Q is given by:

$$D_{KL}(P||Q) = \sum_{x\in X} p(x) log \left(\frac{P(x)}{Q(x)} \right) \tag{2.25}$$

KL divergence is also called relative diversity. Cross entropy measures the total entropy between distributions. It is important to not that cross entropy and joint

entropy are very different terms. Cross entropy of the distribution q relative to p is given by:

$$\begin{aligned} \mathcal{H}(p,q) &= \mathcal{H}(p) + D_{KL}(p||q) \\ &= -\sum_{x \in X} p(x) log q(x) \end{aligned} \tag{2.26}$$

Reinforcement learning algorithms have benefited greatly from concepts from information theory. Entropy regularization (Williams and Peng 1991; Mnih et al. 2016) have been used to increase the stochasticity of policies. Increasing the entropy of the policy promotes exploration and selection of diverse actions fairly. Entropy regularization contributes significantly in the overall performance of the algorithm especially in non-linear function approximation setting. Maximum entropy model-free methods have produced state of the art results on continuous control tasks as well as real-world robotic control (Haarnoja et al. 2017; Haarnoja et al. 2018). Maximum entropy frameworks in a hierarchical reinforcement learning framework generated diverse skills in the absence of task related reward (Eysenbach et al. 2018). Using the learned diverse skills as a prior was shown to be useful to learn challenging continuous control tasks. Entropy-based methods in policy gradient methods reduces the likelihood of the policy to get stuck in local optimum.

In this thesis, we use the properties stated above to incorporate the notion of *diversity in options* in the learning framework. We show that diversity motivated learning improves the exploration as well as overall performance of the algorithm. We also use the diversity term in defining an option's abstraction to exploit diversity further. We empirically show how diversity plays a role in learning useful and stable options. We also show how diversity in options helps identify and learn events faster over an agent's trajectories.

2.5 Deep Reinforcement Learning

Over the years, reinforcement learning has become increasingly popular due to its success in solving challenging sequential decision making tasks. Emergence of deep learning allowed learning representations from complex high-dimensional input data such as images made up of thousands of pixels. With the aid of deep learning, RL has been able to achieve learn policies in a way similar to the way humans learn to perform decision making tasks. Certain deep RL algorithms have shown super-human performance in playing Atari games from pixels (Mnih et al. 2015). Deep RL is the key to bring RL techniques to the real-world applications such as robotic control, self-driving automobiles, drug discovery, finance and more. Despite the endless learning possibilities that deep learning brought to the field, deep RL still faces some important challenges. Scaling RL to complex tasks requires learning good generalized policies which which is able to handle perturbations and out of context data better. Learning representations from high dimensional data requires a lot of data, which is often too expensive to generate. It is hence crucial to develop sample efficient algorithms which can still achieve acceptable performance.

The deep Q-network(DQN) algorithm (Mnih et al. 2015) demonstrated strong performance in a model-free online setting solely learning from pixel bases images. Convolution neural networks (CNN) are used to learn a representation from the observations. DQN used a replay buffer to store experience from past trajectories. The experience was collected using an ϵ-greedy policy. To reduce variance, a mini-batch populated uniformly from the replay buffer was used for updates instead of using values from a single transition. DQN achieved super-human performance on several Atari learning environments. Policy gradient methods which optimize a policy using the maximum expected return objective us a first order method which approximates the function to be quite smooth which is often not the case. Taking large steps in the direction of maximizing returns can cause instability and hurt performance. Trust Region opti-

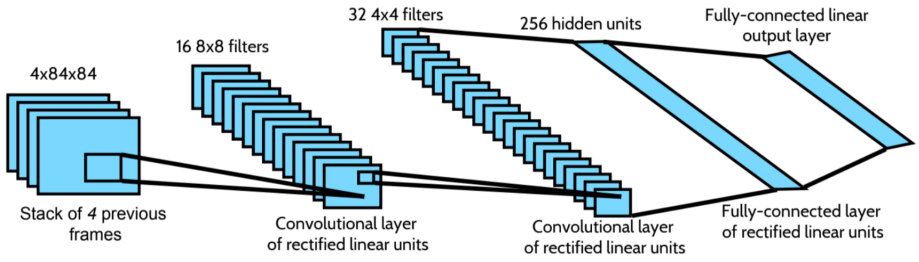


Figure 2.6: DQN Architecture (Mnih et al. 2015)

mization methods aim at improving the policy in a controlled way. The changes in the policy are restricted using the KL-divergence between the action distributions. Bounding the size of policy update also enforces bounds in changes in state distributions thereby guaranteeing policy improvement. TRPO (Schulman et al. 2015) is one of the most popular algorithms which uses constrained updates. The TRPO policy update is given by:

$$\max_{\Delta\theta} \mathbb{E}_{s\sim\rho^{\pi_\theta},a\sim\pi_\theta}\left[\frac{\pi_{\theta+\Delta\theta}(s,a)}{\pi_\theta(s,a)}A^{\pi_\theta}(s,a)\right]$$
$$\text{subject to } \mathbb{E}\left[D_{KL}\left(\pi_\theta(s,\cdot)||\pi_{w+\Delta w}(s,\cdot)\right)\right] \leq \delta) \tag{2.27}$$

where $\delta \in \mathbb{R}$ is a hyper-parameter and A^π is the advantage function. TRPO uses a conjugate gradient with KL constraint to optimize the objective function.

2.5.1 Proximal Policy Optimization Algorithms

Although TRPO emphasises the benefits of using second order methods for policy optimizations, second order methods are naturally more complicated. TRPO is also quite sensitive to architectures which include noise or parameter sharing. To alleviate these problems while maintaining the reliability of of TRPO, Proximal Policy Optimization (PPO) algorithms were introduced (Schulman et al. 2017). Instead of using the KL constraint, PPO uses a penalty in the form of clipped probability ratios which forms a pessimistic estimate of the policy's performance. PPO is an on-policy algorithm where the same policy is used to generate samples as well as for policy

improvements. The surrogate objective for conservative policy iteration (CPI) used in TRPO is:

$$L^{CPI}(\theta) = \mathbb{E}\left[\frac{\pi_\theta(a_t, s_t)}{\pi_{\theta_{old}}(a_t, s_t)} A_t\right] = \mathbb{E}\left[r_t(\theta) A_t\right] \tag{2.28}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t, s_t)}{\pi_{\theta_{old}}(a_t, s_t)}$.

The objective used for PPO is:

$$L^{clip}(\theta) = \mathbb{E}_t\left[\min(r_t(\theta)A_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)\right] \tag{2.29}$$

where ϵ is a hyperparameter. The second term in the objective discourages r_t from moving outside the interval $[1-\epsilon, 1+\epsilon]$. Taking the minimum of the clipped and unclipped objective ensures the final pbjective is a lower bound or pessimistic with respect to the unclipped objective. PPO methods are first order methods, making them simper and easier to implement than TRPO. PPO also shows better sample complexity when tested empirically. On a wide suite of tasks including continuous control tasks implemented in Mujoco and discrete control tasks from Atari learning environment, PPO achieves state of the art performance. We use PPO as a baseline for all out experiments in this thesis.

2.5.2 Proximal Policy Option-Critic (PPOC)

Learning options end-to-end for non-linear function approximation case not only improves exploration and performance, but also interpretability of agent's learning (Bacon, Harb, and Precup 2017; Harb et al. 2018). In section 2.5.1 we discuss the benefits of proximal policy optimization algorithms for model-free reinforcement learning. Proximal Policy Option-Critic (PPOC) (Klissarov et al. 2017) uses the PPO objective for intra-option policy updates. PPOC is capable of learning options end-to-end on a suite of continuous control tasks using the Mujoco simulator. PPOC uses a network for value function as well as termination function and another network for intra-option policy updates as well as policy over option updates. To prolong degeneration, besides

using the deliberation cost (Harb et al. 2018), PPOC reuses samples of options which are inadequately selected. The option improvement updates in PPOC are defined as:

Intra-option policy update:

$$\theta_\pi \leftarrow \theta_\pi + \alpha_{\theta_\pi} \frac{\partial L_t(\theta)^{PPO}}{\partial \theta_\pi} \tag{2.30}$$

where $L_t(\theta)^{PPO}$ is the PPO objective.

Termination function update:

$$\theta_\beta \leftarrow \theta_\beta + \alpha_{\theta_\beta} \frac{\partial \beta(s_t)}{\partial \theta_\beta} \left(A(s_t, o_t) + \eta\right) \tag{2.31}$$

where η is the deliberation cost and $(A(s_t, o_t)$ is the advantage function.

Policy over option update:

$$\theta_\Omega \leftarrow \theta_\Omega + \alpha_{\theta_\Omega} \frac{\partial \Omega(o_t|s_t)}{\partial \theta_\Omega} \left(A(s_t, o_t)\right) \tag{2.32}$$

Critic update:

$$\theta_w \leftarrow \theta_w + \alpha_{\theta_w} \frac{\partial (G_t - Q_\pi(s_t, a_a)^2)}{\partial \theta_w} \left(A(s_t, o_t)\right) \tag{2.33}$$

where G_t is the return.

PPOC outperformed PPO on several standard Mujoco tasks with a learning horizon of 1 million steps showing better sample efficiency. PPOC also demonstrated significantly better performance on transfer tasks. Benefits of learning options are can be better appreciated in certain tasks where a hierarchical architecture can be exploited. For our empirical results we use PPOC as a baseline. We also incorporate our algorithm within the PPOC code-base.

3

Intrinsically Motivate Diversity

Option-critic is one of the most popular frameworks used to learn options end-to-end. The algorithm learns options solely using rewards explicitly received from the environment which are used to optimize expected return. This however, does not incentivize options to learn useful, desirable properties such as *diversity in options*, often leading to different options learning similar behavior. An option set comprising of similar options would imply similar decision sequences irrespective of which option is selected. A diverse option set can be useful in the agent's exploration as each option can specialize to different state space thereby effectively decomposing the problem's complexity. Diversity can also increase the stochasticity of the policy over options promoting options to remain relevant and useful to the task. In this chapter, we draw inspiration from intrinsic motivation to develop a pseudo-reward complementing the task reward which encourages options to maximize diversity. We also show how the maximum diversity objective improves exploration as well as overall performance on several continuous control tasks.

3.1 Encourage Diversity While Learning

A challenge in reinforcement learning is to learn a holistic policy where the agent seeks to acquire useful traits while optimizing returns to improve performance, robustness and interpretibility. A good reward function, can capture more than just information required to perform the task. Engineering an appropriate reward signal while designing environments is very challenging, as in classical reinforcement learning the agent only seeks to maximize expected return. Humans often adopt strategies which offer intrinsic benefits such as for enjoyment or comfort. These abstract feelings aren't really useful in performing everyday tasks, still they play an important role in how humans make decisions. In reinforcement learning, intrinsic motivation deals with methods where the agent acknowledges actions which increase intrinsic gains such as exploration or safety. In this thesis, we derive a term which represents the diversity of options at a given state, and incorporate it within the reward function with the intention of increasing overall diversity in options.

While most relevant literature on learning diverse options (Gregor, Rezende, and Wierstra 2016; Eysenbach et al. 2018; Harutyunyan et al. 2019) use states to distinguish and localize options, we instead look directly at an option's behavior to assess its diversity. This idea is well suited when all options are available everywhere, when the state information is imperfect (for example, because the latent representation of the state is still being learned), and when the agent aims to transfer knowledge across tasks. The accessibility to all options at any given state allows the agent to effectively reuse specialized options (Bacon, Harb, and Precup 2017). Consider a locomotion task where the agent is required to leap over several hurdles in its trajectory. An option specialized for leaping over hurdles can be selected whenever the agent encounters a hurdle throughout its trajectory as its initiation is never restricted. Hence, reusabil-

ity of options require a relaxed initiation set in model-free learning. Reusability of specialized options have also been useful in transfer tasks. An example of a transfer task could be displacing the hurdles from the above example after a few episodes of training. We study reusability and transfer characteristics of options in chapter 4. Unlike most diversity based approaches where skills are learned in an unsupervised setting (Gregor, Rezende, and Wierstra 2016; Eysenbach et al. 2018), our approach is capable of learning the task simultaneously while encouraging diverse option policies. For simplicity of exposition, we use two options in our notation in this chapter; however the approach can be easily extended to any number of options. An empirical study with varying number of options are presented in Section 4.5.

We construct our pseudo reward function using concepts from information theory. Maximizing the entropy of a policy prevents the policy from quickly falling into a local optimum and has been shown to have substantial improvements in exploration and robustness (Williams and Peng 1991; Mnih et al. 2016; Haarnoja et al. 2018). We maximize the entropy $\mathcal{H}(A^{\pi_1} \mid S)$ and $\mathcal{H}(A^{\pi_2} \mid S)$ where $\mathcal{H}$ is the Shannon entropy computed with base e and A represents respective action distributions. Since we want want different options behave differently from each other at a given state, we maximize the divergence between their action distributions $\mathcal{H}(A^{\pi_2}; A^{\pi_2} \mid S)$. This aligns with our motivation that skill discrimination should rely on actions. Lastly, we seek to maximize the stochasticity of the policy over options $\mathcal{H}(O^{\pi_\Omega} \mid S)$ to explore all available options at S. Combining all the above terms, we get the following pseudo reward $\mathcal{R}_{bonus}$:

$$\mathcal{R}_{bonus} = \mathcal{H}(A^{\pi_1} \mid S) + \mathcal{H}(A^{\pi_2} \mid S) + \mathcal{H}(O^{\pi_\Omega} \mid S) + \mathcal{H}(A^{\pi_2}, A^{\pi_2} \mid S) \tag{3.1}$$

The first three terms in Eq 3.1 aim to increase the stochasticity of intra-option policies as well as policy over options while the fourth term encourages overall diversity

in the option set. Since we use entropy regularization for intra-option policies as well as policy over options updates in all our experiments, we omit $\mathcal{H}(A^{\pi_1} \mid S), \mathcal{H}(A^{\pi_2} \mid S)$ and $\mathcal{H}(O^{\pi_\Omega} \mid S)$ from our pseudo reward. Implementation details are provided in Appendix A.

We incorporate this objective within the standard RL framework by augmenting the reward function to include the pseudo reward bonus from Eq (3.1):

$$\mathcal{R}_{aug}(S_t, A_t) = (1 - \tau)\mathcal{R}(S_t, A_t) + \tau\mathcal{R}_{bonus}(S_t) \tag{3.2}$$

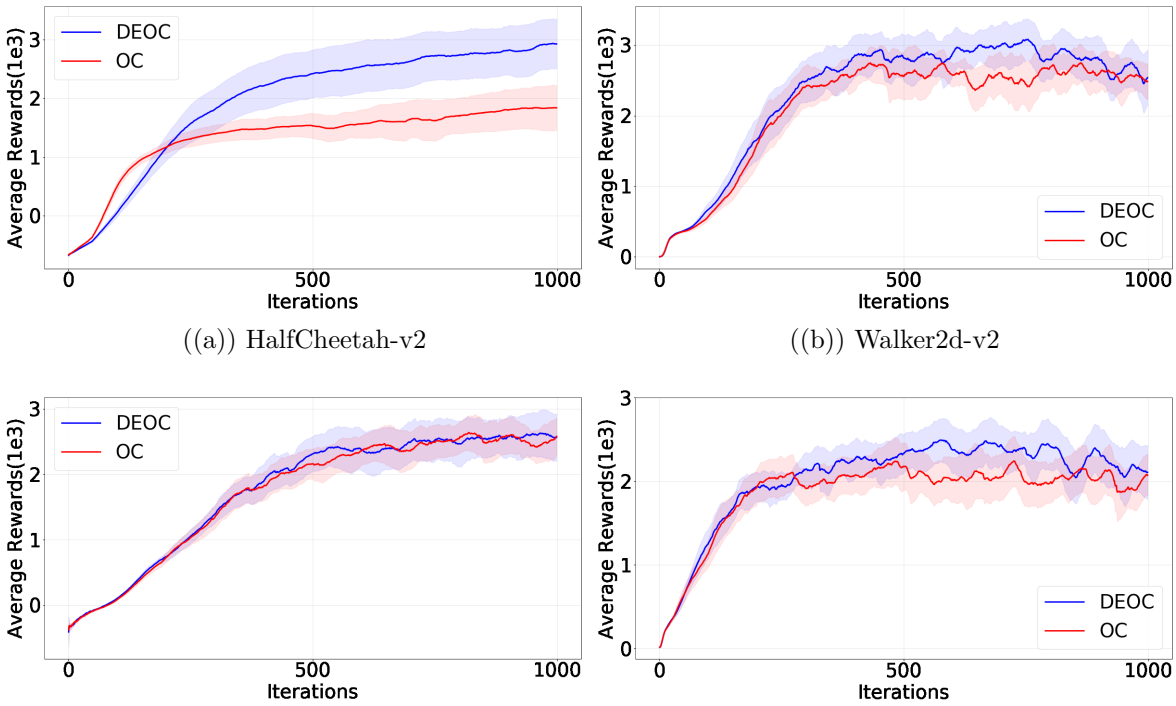
where $\mathcal{R}$ is the task reward function, $\mathcal{R}_{bonus}$ is the pseudo reward bonus which stems from Eq (3.1) and τ is a hyper-parameter which controls relative importance of the diversity term against the reward. As the trade-off, τ increases, the agent increases its priority of discriminating options over learning the task. The proposed reward augmentation yields the maximum diversity objective. The standard RL objective can be recovered in the limit as $\tau \to 0$.

3.2 Experiments

To study the effects of augmenting the reward with a bonus intended for promoting diverse option behaviors, we test our algorithm, Diversity Enriched Option-Critic (DEOC), against Option-Critic (OC) on several classic continuous control tasks implemented in Mujoco (Todorov, Erez, and Tassa 2012).

We use the Proximal Policy Option-Critic (PPOC) (Dhariwal et al. 2017; Klissarov et al. 2017) codebase to build our algorithms as well as for OC and PPO baselines. The reward augmentation is shown in Algorithm 1. We use the same hyper-parameter settings across all 20 seeds in all our experiments throughout the thesis to test stability. Since it is imperative that the task reward is not outweighed by $\mathcal{R}_{bonus}$, we scale down $\mathcal{R}_{bonus}$ by a factor of five (which is selected using a coarse hyper-parameter tuning) for all non-linear function approximation tasks. This ensures the agent always

prioritizes learning the task over diversifying options. We plot the results for 1000 iterations to contrast differences in sample efficiency. Results with longer horizons are presented in the chapter 4 where we test the stability of all algorithms.



((a)) HalfCheetah-v2

((b)) Walker2d-v2

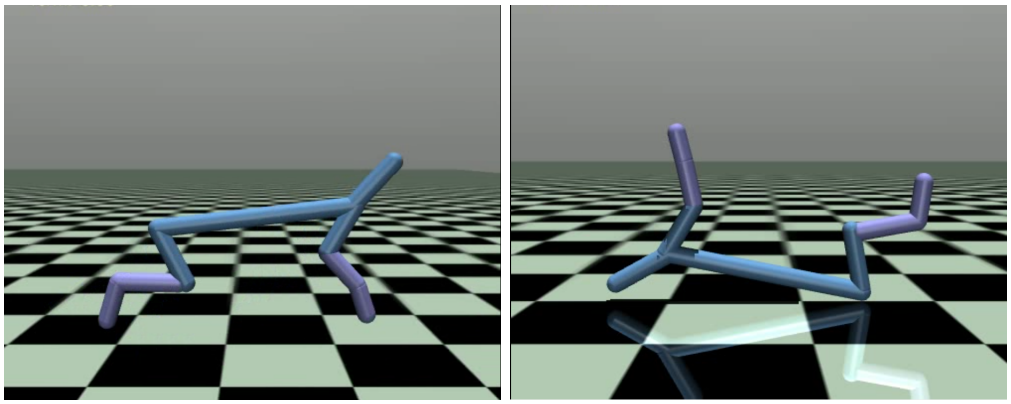
((c)) Ant-v2

((d)) Hopper-v2

Figure 3.1: **Empirical Results of DEOC compared against OC** for 2 million steps (1 Iteration = 2048 steps). Trade-off value (τ) is 0.7 for HalfCheetah-v2 (Fig 3.1(a) while τ is 0.2 for remaining tasks. Plot is an average of 20 independent runs.

Figure 3.1 show that introducing the auxiliary reward bonus improves sample efficiency as well as performance. The most significant impact was noticed in HalfCheeetah-v2 where a good exploration strategy plays a huge impact. OC learns a gait where the agent first flips over and slides on its back. Through this, the agent avoids tripping and losing balance. Unfortunately, not only does this gait limit the agent's ability to progressively run faster, it also limits its adaptability to changes or obstacles in the environment. For example, sliding on the back makes it difficult to leap over hurdles and obstacles which may be present in the agent's trajectory. HalfCheetah-v2 presents itself as a limitation of Proximal Policy based algorithms. Learning to exploit diverse skills prevents the agent to prematurely adopt the first viable strategy it happens

upon by increasing exploration and encouraging the agent to further explore all skills. DEOC manages to almost always run upright, achieving significantly higher speeds and returns. From Fig 3.1(a) we can observe DEOC outperforms OC by almost 2000 cumulative rewards. Figure 3.2 shows a screenshot of the gaits learned by both DEOC and OC. Videos of all our experiments are provided in our website [1]. Implementation details are provided in Appendix A.1.3



((a)) Diversity Enriched Option-Critic (DEOC)

((b)) Option-Critic (OC)

Figure 3.2: **Illustrations showing the gaits learned by DEOC and Option-Critic (OC).** Unlike OC where the agent flips over and slides on its back, DEOC almost always learns to run upright.

[1] https://sites.google.com/view/deoc/home

4

Encourage Diversity in Termination

In chapter 3, we empirically demonstrate how encouraging diversity in option policies while learning, significantly improves exploration and performance of option-critic.

However, unlike primitive action policies where all actions are available at every step, options execute for variable time steps until a termination condition is met, during which, all other options remain dormant. Due to this, the maximum entropy objective fails to be as effective with options as with primitive action policies. Although having options terminate at every time step may solve this problem, it renders the use of options moot. Additionally, option-critic's termination function solely validates the best option, suppressing other potentially viable options which may also lead to near-optimum behavior. As a consequence, at a given state, only the best current option gets updated, eventually leading to a single option dominating the entire task. Diverse skills can only be useful if all available skills are justly explored and selected. If an option remains dominant throughout the episode without terminating, the agent cannot reap the benefits of temporal abstraction nor learning diverse skills. Noise in value estimates or state representations may also cause an option to terminate and consequently lead to the selection of a sub-optimal option. We discuss how selecting a sub-optimal option around "vulnerable" states can be catastrophic and also severely hurt performance. In our case, despite $\mathcal{R}_{bonus}(s, a)$ encouraging diverse options, option-critic's termination function prevents exploiting this diversity due to

inadequate exploration of all relevant options.

In this chapter we present a novel termination objective which no longer aims to maximize the expected discounted returns, but focuses on the option's behavior and identifying states where options are distinct, while still being relevant to the task. Our proposed objective encourages all options available at a given state to be explored so long as they exhibit diverse behavior. Not only does the algorithm achieve significant improvements in performance and robustness, it offers better interpretibility as to the behavior of each option.

The termination component of the option's framework signifies the purview of an option to the task. There has been a lot of interest in defining the termination condition such that options can learn useful strategic behavior (Sutton, Precup, and Singh 1999; Bacon 2013; Bacon, Harb, and Precup 2017; Bacon 2018; Harutyunyan et al. 2019). However, understanding what makes an option useful in a context of learning a task is quite subjective, which is why this this remains an ongoing challenge. Option-critic derived the termination gradient (Eq (2.17)) from the maximum expected return objective. The termination gradient theorem indicates that an option may terminate while its value is sub-optimal as compared to another. However, since primitive actions are sufficient for solving any MDP, options begin defaulting to primitive actions. This degeneration in option-critic algorithms can be explained through the advantage function in the objective. The termination likelihood of an option only increases while the value of the current option is sub-optimal when compared to the value of another. A consequence that follows is that at a given state the *worse* option is quickly suppressed without adequate exploration while the best available option keeps improving. Eventually, this often leads to a single option dominating the entire task. If the agent ceases to switch options altogether, the need for temporal abstraction is lost along with its advantages in exploration, robustness, transfer and overall performance.

Noise in value estimates or state representations may also cause an option to

terminate and consequently lead to the selection of a sub-optimal option. Due to lack of sufficient exploration of the sub-optimal option, selecting it around certain vulnerable states may severely hurt performance. Vulnerable states in this context refers to states where taking an sub-optimal action can lead to catastrophic outcomes and the end of the episode. Consider the standard Hopper-v2 experiment evaluated in chapter 3. While hopping, it becomes vital to ensure perfect balance while landing. Hence, states immediately prior to landing are very sensitive to the agent's actions, where even a slight misalignment can cause the episode to terminate. The agent may not be able to afford selecting a sub-optimal option at these states especially when the sub-optimal option hasn't been explored and updated adequately around these states. In this section we present a termination objective which, unlike the termination gradient theorem, no longer aims to maximize the expected discounted returns, but focuses on option's behavior and identifying states where options are capable of behaving quite distinctly while still being relevant to the task.

Owing to the points discussed above, we build our objective function to oblige the following two conditions:

- **Options should identify and terminate in states where options tend to grow diverse.** Most tasks have certain critical states which can inspire diverse behaviors. For example in the classic four-rooms task (Sutton, Precup, and Singh 1999), these can be the hallways around which different options can easily adopt different navigation strategies such as entering the room or turning back, depending on the location of the goal state. (Bacon, Harb, and Precup 2017) have demonstrated terminations localized around hallways leads to better performance in transfer settings. Allowance of the agent's ability to sustain diverse skills can be interpreted as its ability to decompose the breadth of the task using diverse yet relevant options. This property is very useful to scale algorithms to more complex tasks.

- **The diversity metric used in the termination objective should capture the diversity relative to other states in the sampled trajectories.** We wish terminations to focus on states where options are most diverse, relative to other observed states. This makes sense as diversity in options can be better exploited for exploration and stability around these states. In the context of using skills to decompose the scope of the problem, identifying states which inspire the most diverse options can help the agent exploit options to learn the most challenging sections of the tasks. Offering more than one useful options around these states not only increases robustness and performance, it reduces the sample complexity when compared to exploring all available skills on all states in its trajectory. In the following sections we also discuss the potential of this property to identify events or changes in the trajectory.

The termination objective we maximize becomes:

$$L(\theta_\beta) = \mathbb{E}\Big[\beta(S_t, O_t)\mathcal{D}(S_t)\Big] \tag{4.1}$$

The term $\mathcal{D}(S_t)$ indicates the relative diversity of options at a given state. To compute $\mathcal{D}(S_t)$, we use samples of auxiliary reward ($\mathcal{R}_{bonus}(S_t)$) defined in Eq (3.1) which is a measure of how diverse options are at a given state. $\mathcal{R}_{bonus}(S_t)$ however, is a positive term, which would consequently always increase the termination likelihood for any state. We mitigate this by standardizing (with a mean (μ) of zero, and standard deviation (σ) of one), the $\mathcal{R}_{bonus}(S_t)$ samples collected in the buffer, to obtain $\mathcal{D}(S_t)$.

$$\mathcal{D}(S_t) = \frac{\mathcal{R}_{bonus}(S_t) - \mu_{\mathcal{R}_{bonus}}}{\sigma_{\mathcal{R}_{bonus}}} \tag{4.2}$$

Not only does this solve the issue of constant termination at all states, we also scale the updates relative to the diversity values of other states in the buffer. Terminating while options are most diverse encourages both options to be selected fairly and explored by the policy over options. Note that as with $\mathcal{R}_{bonus}(S_t)$, $\mathcal{D}(S_t)$ is independent of the termination parameters.

Hierarchical frameworks are believed to be the key behind scaling reinforcement learning to exceedingly challenging tasks due to its ability to decompose the problem space using specialized skills. The options framework has the potential to uphold this property so long as options remain relevant, mutually distinct and specialized. Our diversity motivated termination objective identifies regions where highest relative diversity can be sustained and learns specialized useful options to achieve better control and performance. An added advantage of using relative diversity is the agent's ability to respond to events or obstacles in its trajectory. An example of such an event could be the presence of a hurdle in a locomotion task. Such events mark some of the most sensitive and vulnerable states in the environment. Relative diversity $\mathcal{D}(S_t)$, in our objective, is capable of identifying states around the events causing both options to collectively explore and learn the event.

4.1 Empirical Evaluation

We evaluate the effects of the new termination objective on several tasks to test its performance and stability. We introduce Termination-DEOC (TDEOC) algorithm presented in Algorithm 1. Detailed description of all the tasks are provided in Appendix E.

4.1.1 Tabular Four-rooms Navigation Task

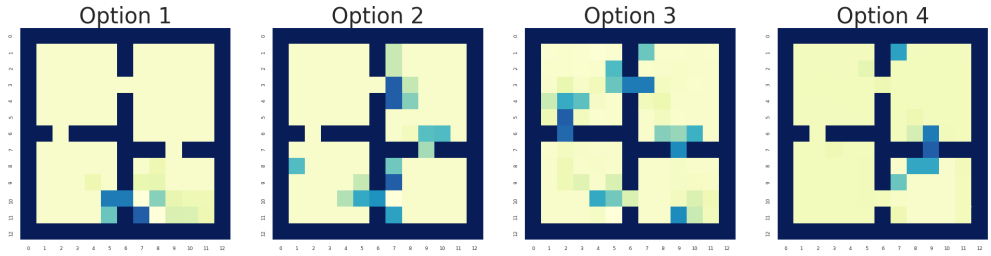
We first test our algorithm TDEOC, on the classic four-rooms navigation task (Sutton, Precup, and Singh 1999) where Bacon, Harb, and Precup 2017 demonstrated the model-free learning and transfer capabilities of options against primitive action frameworks. For our four-rooms experiments, we reuse the implementations by Bacon, Harb, and Precup 2017. Initially the goal is located in the east hallway and the agent starts at a uniformly sampled state. After 1000 episodes, the goal state is moved to a random location in the lower right room. The goal state yields a reward

Algorithm 1 Termination-DEOC (TDEOC) algorithm with tabular intra-option Q-Learning

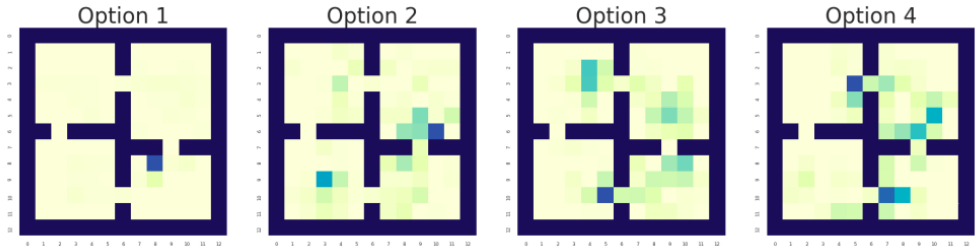
Initialize policy over options (π_Ω)
Initialize intra-option policy (π_o)
Initialize termination function (β_o)
$s_t \leftarrow s_0$
Choose o_t according to $\pi_\Omega(o_t|s_t)$
repeat
  Choose a_t according to $\pi_o(a_t|s_t)$
  Take action a_t in s_t and observe s_{t+1} and r_t
  Compute $r_{bonus}(s_t)$
  $r'_t = (1-\tau)r_t + \tau\, r_{bonus}(s_t)$
  if o_t terminates in s_{t+1} **then**
    Choose new o_{t+1} according to $\pi_\Omega(\cdot|s_{t+1})$
  else
    $o_{t+1} = o_t$
  end if
  $D(s_t) \leftarrow$ Standardized samples of $r_{bonus}(s_t)$.
  Options Evaluation:
  $\delta \leftarrow r'_t$ - $Q_U(s_t, o_t, a_t)$
  $\delta \leftarrow \delta + \gamma(1 - \beta_{o_t}(s_{t+1}))Q_\Omega(s_{t+1}, o_t) +$
    $\gamma\beta_{o_t}(s_{t+1})max_{o_{t+1}}Q_\Omega(s_{t+1}, o_{t+1})$
  $Q_U(s_t, o_t, a_t) \leftarrow Q_U(s_t, o_t, a_t) + \alpha\delta$
  Options Improvement:
  $\theta_\pi \leftarrow \theta_\pi + \alpha_{\theta_\pi}\frac{\partial log\pi_{o_t}(a_t|s_t)}{\partial\theta}Q_U(s_t, o_t, a_t)$
  $\theta_\beta \leftarrow \theta_\beta + \alpha_{\theta_\beta}\frac{\partial\beta_{o_t}(s_{t+1})}{\partial\nu}\ \mathcal{D}(s_{t+1})$
until s_{t+1} is terminal

of +1 while all other states produce no rewards. The action the agent selects can fail with a probability of one third ($\frac{1}{3}$) and instead a random action is taken. Augmenting the reward with $\mathcal{R}_{bonus}$ for tasks with very sparse rewards can cause the agent to prioritize diversifying options over learning the task. To mitigate this, we avoid the reward augmentation step in the current four-rooms task. The relative diversity term $\mathcal{D}(S_t)$ requires standardizing $\mathcal{R}_{bonus}$ samples at every time step. However, in a tabular setting using a buffer to record samples at every step severely affects the time complexity of the algorithm. Instead of standardizing the diversity values ($\mathcal{R}_{bonus}$) at every step, we update a moving sum of all values observed in the current run and center $\mathcal{R}_{bonus}$ samples around the moving mean instead. Not only does this offer bet-

ter time complexity than the alternative, it maintains the paradigm of the proposed termination objective. Implementation details are provided in Appendix A.1.2.



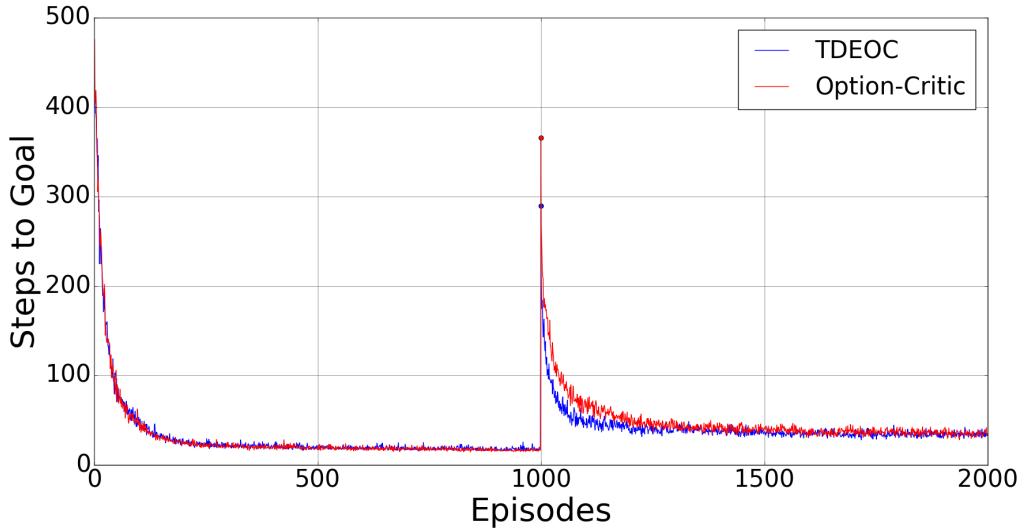
((a)) Termination-DEOC (TDEOC)



((b)) Option-Critic (OC)

Figure 4.1: **Visualization of Terminations for different options** after 1000 episodes. Darker colors correspond to higher termination likelihood. Both TDEOC and OC show higher terminations around hallways.

The termination probabilities of both Option-Critic (OC) and TDEOC are visualized in Fig 4.1. The darker colors represents higher termination probabilities while the darkest colors represent the walls of the environment. TDEOC identifies the hallways as the '*bottleneck*' states. These states represent regions where options can grow most diverse relative to other states observed by the agent. Hallways seem to be the most intuitive regions to define sub-goals which both Option-Critic as well as TDEOC seek to define options' terminations. Fig 4.2 compares the learning and transfer performanaces of OC and TDEOC. Both TDEOC and OC have nearly the same learning rate for the first 1000 episodes. Upon changing the goal state, we notice TDEOC recovers faster than OC by almost 70 time steps. Such a difference is quite drastic in tabular evaluations between algorithms with near identical learning frameworks. Not only does TDEOC recover faster than OC, it also exhibits lower variance, indi-



((a)) Termination-DEOC (TDEOC) VS OC

Figure 4.2: **Four-rooms transfer experiment with four options**. After 1000 episodes, the goal state, is moved from the east hallway to a random location in the south east room. TDEOC recovers faster than OC with a difference of almost 70 steps when the task is changed. Each line is averaged over 300 runs.

cating a more confident strategy. In order to achieve fair results, hyper-parameters of both algorithms are tuned separately and the best configurations are used to plot the results in Fig 4.2.

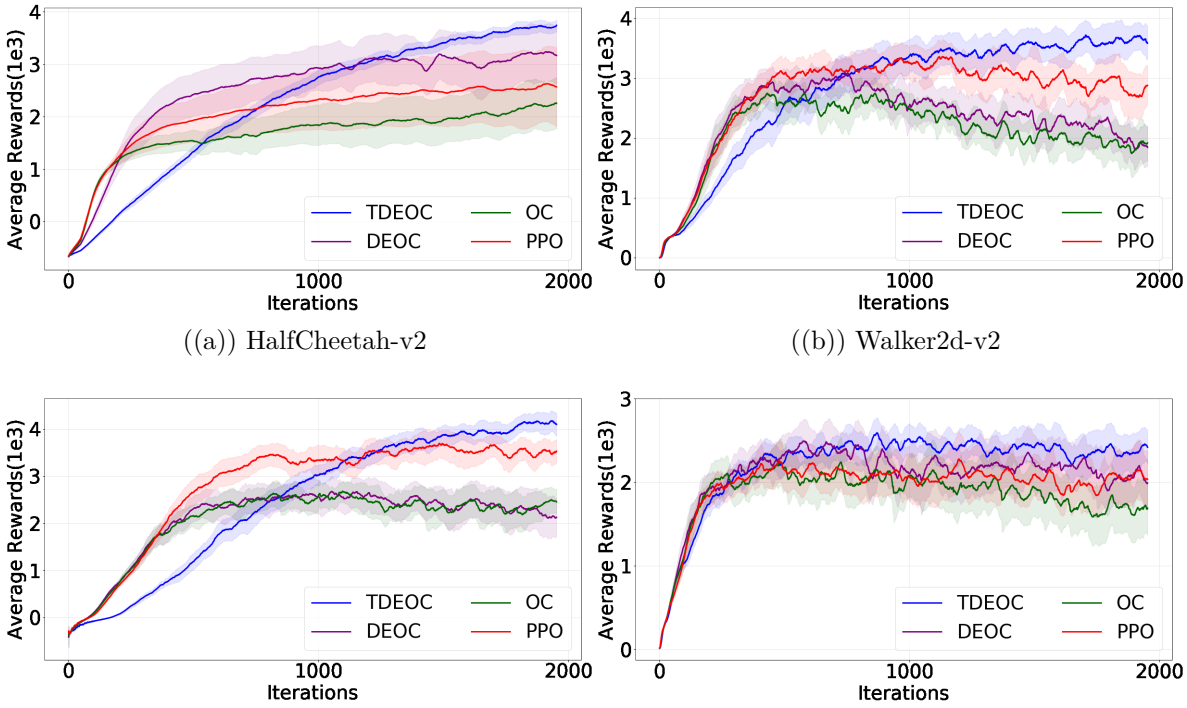
4.1.2 Continuous Control Tasks

Next, we show the advantages of a diversity-targeted termination in the non-linear function approximation setting using standard Mujoco (Todorov, Erez, and Tassa 2012) tasks. Since PPOC uses a replay buffer to collect samples from the current policy we can easily standardize the $\mathcal{R}_{bonus}$ samples. We evaluate the algorithms on longer learning horizons (than in Fig 3.1) to contrast stability along with overall performance. The average results from 20 runs are plotted (Fig 4.3). Details about the implementation and the hyper-parameters are provided in Appendix A.1.3.

We evaluate the performance of TDEOC against Option-Critic (OC), PPO (Schulman et al. 2017) and the DEOC algorithm defined in chapter 3. Not only does TDEOC

significantly outperform OC and DEOC, it also exceeds the performance of PPO which boasts state-of-the-art performance with on-policy learning on most of these tasks. In most of the tasks presented in Fig 4.3, PPO has consistently performed better than OC. However, TDEOC manages to exploit diverse specialized options to decompose the challenging and sensitive regions of the tasks thereby achieving much better performance and significantly lower variance. Environments where better exploration strategies impact performance greatly, such as HalfCheetah-v2, Walker2d-v2 and Ant-v2, TDEOC accumulates as much as 2000 average rewards more than OC. Another interesting observation is that while OC and DEOC quickly stagnate to a sub-optimal solution, TDEOC keeps on improving. We believe the reason for such stagnation is sub-optimal option selection caused by terminating due to noisy value estimations. Learning tasks where the balance is crucial, it can be disastrous to select an option which hasn't been adequately trained. We can observe for continuous control tasks that during the initial stages, one of the options quickly dominates over the entire task preventing the other option to be selected and explored. Consequently, any noise-related option terminations occurring henceforth would most likely lead to the selection of the sub-optimal option and a sub-optimal action which follows. Taking a *bad* action can be catastrophic in states where the '*balance*' is vital. Not only does this phenomenon prevent further improvement, it often negatively affects performance. This can be observed in tasks such as Walker2d-v2 (Fig 4.3(b)) and Hopper-v2 (Fig 4.3(d)). There is a noticeable dip in the performance of OC and DEOC algorithms in the later stages of training. TDEOC on the other hand encourages both options to remain relevant to the task complementing each other to progressively improve performance. PPOC uses entropy regularization for policy over options updates to promote both options to be adequately explored and trained around states of terminations. TDEOC manages to generate diverse yet cooperative option trajectories to stabilize vulnerable regions of the task. This explains why TDEOC manages to handle environment perturbations more robustly. We study the *critical states* which

inspire diversity in section 4.4. Continuously grooming both options to remain relevant and useful impacts sample complexity of the algorithm. We study each option's relevant activity pertaining to the task in section 4.2. Since the termination gradient suppresses the *worse* option from being selected and explored, the dominant option ceases to terminate throughout the episode. This phenomenon is particularly evident in continuous control tasks such as those presented in Fig 4.3. Learning a single policy would naturally prove to be more sample efficient than learning multiple intra-option policies. Due to this, TDEOC exhibits slower rate of learning in most Mujoco tasks during the earlier stages to learning.



((a)) HalfCheetah-v2 ((b)) Walker2d-v2

((c)) Ant-v2 ((d)) Hopper-v2

Figure 4.3: **TDEOC results on standard Mujoco tasks** recorded for four million steps (1 Iteration = 2048 steps). Plots are averaged over 20 independent runs.

Since TDEOC manages to outperform OC and PPO in all the four tasks, we opt to test its capabilities on a more challenging task. We consider the Humanoid-v2 task also implemented in Mujoco. Humanoid-v2 is an exceedingly challenging task which most algorithms fail to learn satisfactorily. The agent is not only required to balance

on two legs, it is also required it to move forward as fast as possible while maintaining this balance. PPO currently produces state-of-the-art results for a model-free on-policy algorithm. We test TDEOC along with OC and DEOC on Humanoid-v2 and the results are presented in Fig 4.4. TDEOC surpasses the performance of OC and DEOC by almost 2500 average rewards. An even more interesting observation is that TDEOC even manages to outperform PPO with a difference of around 800 average rewards.

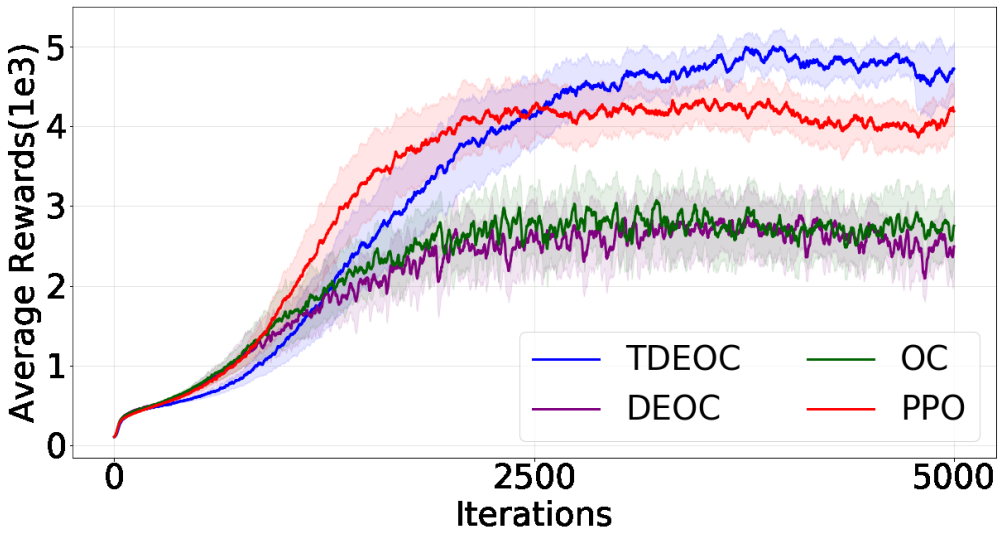


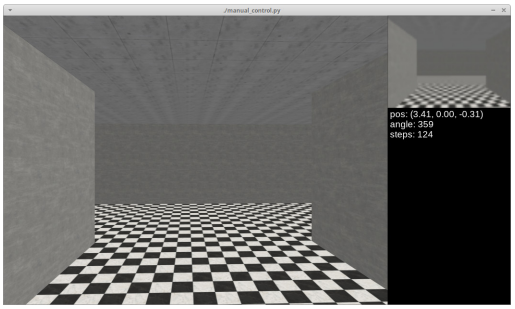
Figure 4.4: **TDEOC results on standard Humanoid-v2** task implemented in Mujoco recorded for ten million steps (1 Iteration = 2048 steps). Plots are averaged over 20 independent runs.

4.1.3 Sparse Reward Tasks

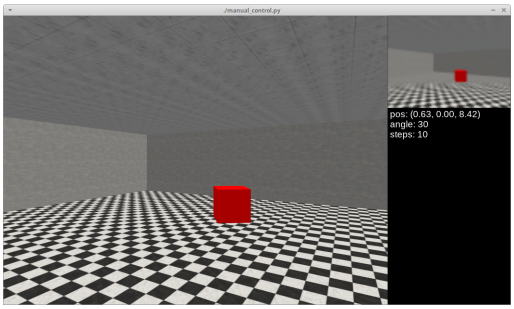
In section 4.1.2, we consider continuous control tasks. In this section, we evaluate TDEOC on discrete 3D control tasks using pixel data stream as input. The tasks are implemented using Miniworld's gym simulation (Chevalier-Boisvert 2018). At any state agent is required to select one of four actions: turn right, turn left, move forward and move backwards. We develop a network to develop state features using a Convolutional Neural Network (CNN) discussed in chapter 2. The network architecture is made to resemble the PPO network for atari environments (Schulman et al. 2017).

We consider the T-Maze, Sidewalk and OneRoom environment from Miniworld. All these environments use a sparse reward signal produced only when the agent happens upon the goal. Due to the presence of sparse reward, as with the four-rooms task, we do not augment the reward with $\mathcal{R}_{bonus}(S_t)$, but still compute it for termination updates. DEOC runs are also omitted due to the same. The OneRoom environment is a

((a)) Sidewalk



((b)) TMaze (Discrete)

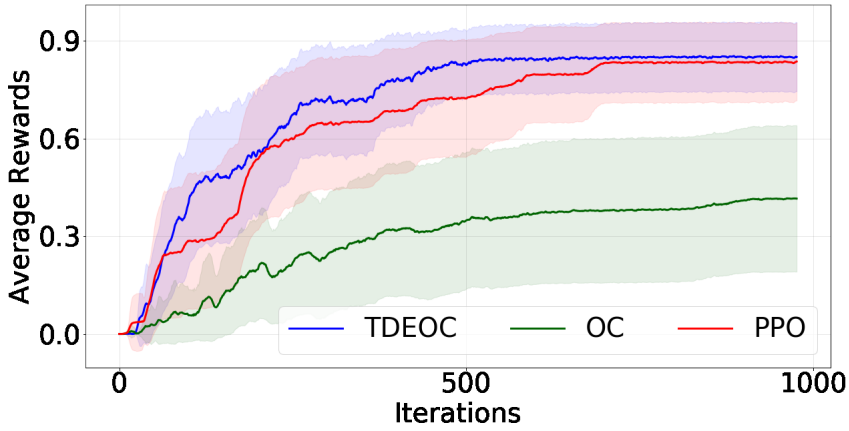


((c)) OneRoom

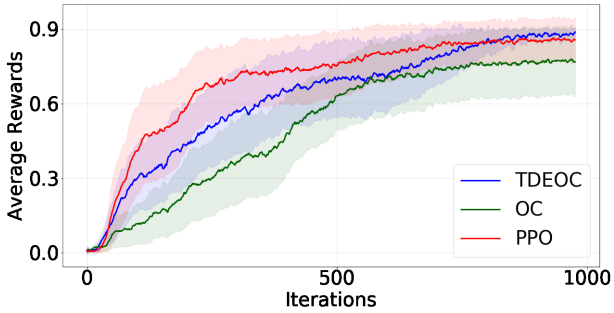
Figure 4.5: Screenshots depicting the Miniworld environments.

closed rectangular room with a goal object placed randomly within it. At the start of the episode the agent is initiated randomly at a location inside the room. The agent requires to scan the room, and once it has the goal object in sight, navigate towards it. All Miniworld environments yield a reward of +1 when the goal is reached and no rewards given for all other states. The T-Maze environment contains a T-junction. The agent is randomly placed at a location in the vertical hallway and the goal in the form of a red box is randomly placed at the either ends of the horizontal hallway. The agent is required to first move towards the end of the vertical hallway, scan for

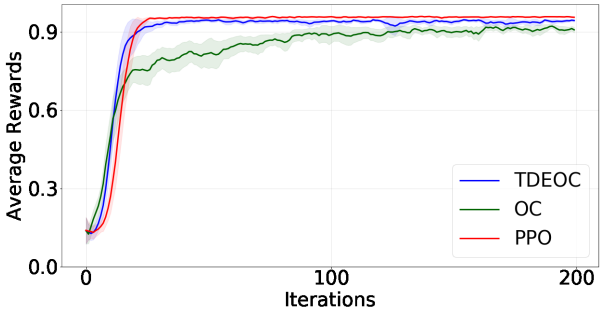
the goal and navigate towards it. The Sidewalk environment consists of a path with one side opening to the road. If the agent moves outside the sidewalk and onto the road, the episode terminates. The goal is placed at the end of the pathway while the agent is initiated at the beginning of the sidewalk. Screenshots of the environments are presented in Fig 4.5



((a)) Sidewalk



((b)) TMaze (Discrete)



((c)) OneRoom

Figure 4.6: **TDEOC results on standard Miniworld tasks** recorded for two million steps except for OneRoom task which is recorded for half a million steps (1 Iteration = 2048 steps). Plots are averaged over 20 independent runs.

Empirical results are presented in Fig 4.6. We observe that while OC stagnates to a sub-par solution for both the tasks, TDEOC manages to learn a better solution faster. Most algorithms find sparse reward problems very difficult to solve owing to the vast multi-step exploration strategy which is only validated by a simple reward, making efficient credit assignment challenging. Being a single policy framework, PPO not only performs better than OC, it also exhibits better sample efficiency. However,

despite carrying the added complexity of learning a hierarchy, TDEOC manages to surpass the performance of PPO in Sidewalk. Achieving good results in Sidewalk relies on the agent's ability to move quickly towards the goal without deviating onto the street. TDEOC has already shown to perform better in tasks consisting of vulnerable and sensitive states which can prompt an episode to terminate. We can also observe significantly lower variance in TDEOC plots when compared to those with OC and PPO. Unlike continuous control tasks, OC in Miniworld environments don't experience the same degeneration. Both options remain relevant throughout the learning cycle. However, TDEOC learns intuitive and interpretable options which explains why TDEOC achieves better performance. We study the option's interpretibility in section 4.4. The implementation details and hyper-parameters are provided in Appendix A.1.3.

4.2 Option Relevance

The ability to decompose the problem space of the task using multiple policies can help scale reinforcement learning to more complex and challenging tasks. We discuss this property briefly in section 4.1.2. Considering the option's framework, each option can be specialized to solve a subset of the task in a way that offers better generalized performance, interpretability and reusability of these options. In the context of model-free learning, learning options end-to-end in this manner is exceedingly challenging. We mention in section 4.1.2 that options often undergo degeneration where the algorithm begins defaulting to primitive actions over temprally-extended actions. In this thesis, we seek to promote every option's relevance towards the task in focus. Option-critic often tends to suppress the worse option quickly such that a single option performs the entire task without termination. The usefulness of temporal abstraction is voided once the agent begins performing the task using a single option. For this reason, it is important to encourage all available options to remain relevant

and useful in the context of performing a task. In this section, we study the effects of our proposed termination objective on each option's relevance.

We first consider the HalfCheetah-v2 task. In section 4.1.2, we study that TDEOC significantly outperforms OC for this particular task. Figure 4.7 plots each option's relevance for both TDEOC and OC. An option's relevance is computed in terms of how many steps that option was active in an iteration or in other words, the number of steps taken by said option in an iteration. The dominant option, which is the most active of the two options is labelled as *Opt1* while the other is names *Opt2*. Figure 4.7 shows that the most active option quickly dominates ceasing to terminate at all. The least active option on the other hand is barely selected. TDEOC on the other hand, shows both options to be much better sampled and explored. The repeated and consistent selection of both options indicates that both options are useful to solve the task. The tiny fluctuations in the plots belonging to OC represent erratic and tiny chances of the least dominant option to be selected. This validates our assumption that wrongful terminations caused by factors such as noisy value estimates or state representations can lead to the selection of the sub-optimal option (Opt2). Selecting Opt2 without adequately exploring and training it can hurt performance significantly.

Since TDEOC encourages diverse options using the intrinsic reward defined in Eq (3.2) and the diversity motivated termination objective defined in Eq (4.1), TDEOC manages to learn specialized options localized near states where options can grow most diverse. As mentioned earlier, the ability to sustain diverse options can be seen as options decomposing the problem space by learning diverse strategies around states causing termination. This ability also contributes towards the improved performance of TDEOC when compared to OC. Figure 4.8 visualizes option relevance of the remaining Mujoco tasks. All the other tasks exhibit the same pattern as that observed in Fig 4.7. In all the other Mujoco experiments OC almost always lets one of the option dominates while the other option is not explored sufficiently. TDEOC on the other hand always allows both options to remain useful. In the Humanoid-v2 task,

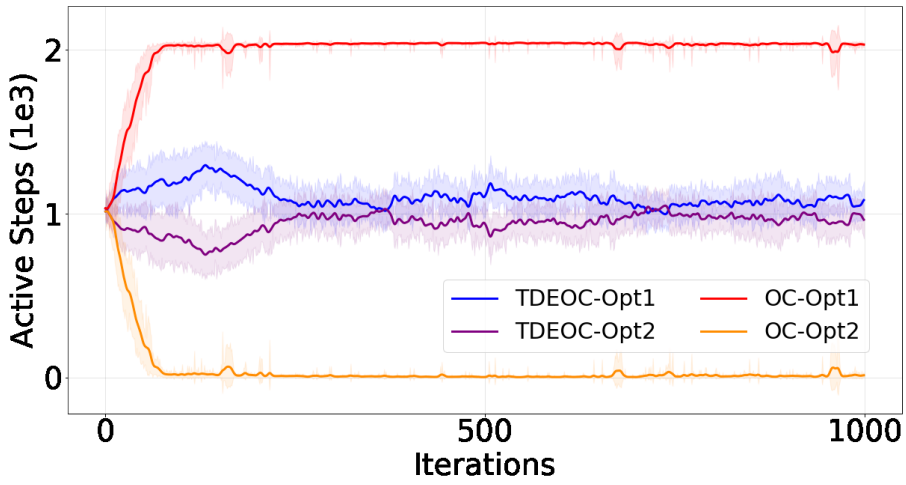
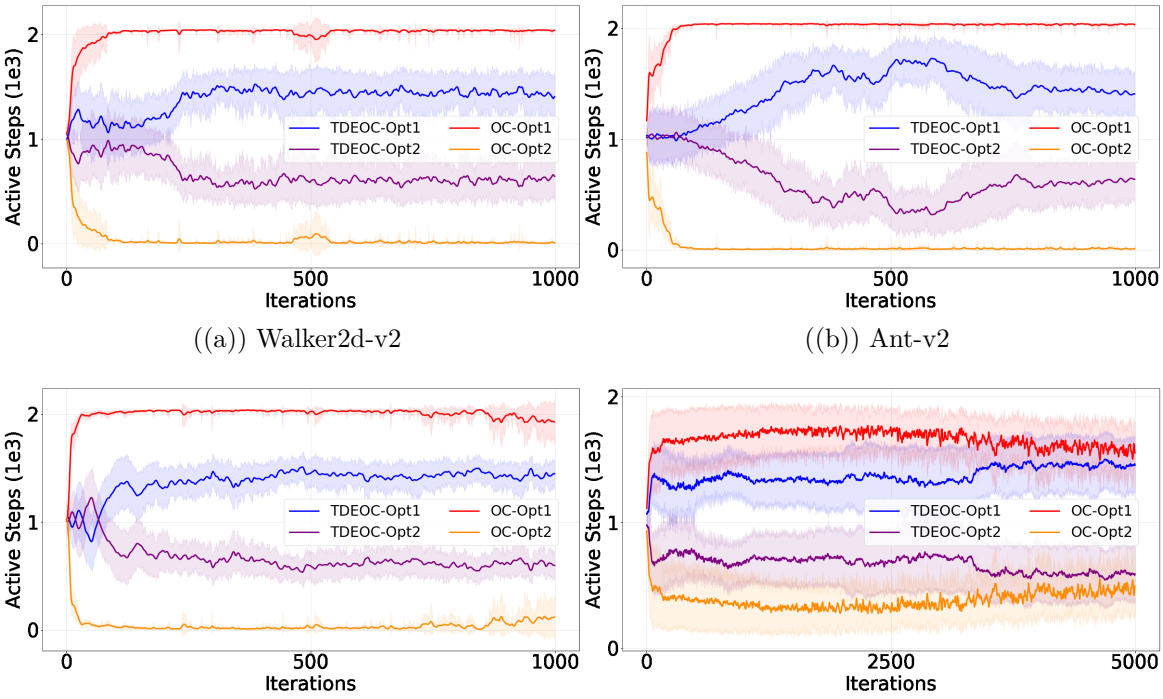


Figure 4.7: **Option activity for standard HalfCheetah-v2** task implemented in Mujoco recorded for two million steps (1 Iteration = 2048 steps). Plots are averaged over 20 independent runs.

OC suggests a more progressive strategy encouraging a more relevant Opt2. However this is most likely caused frequent termination due to high variance and noise due to the added complexity of the task. This reason is validated by the performance curve shown in Fig 4.4. If each option switch contained strategic value, OC would have demonstrated much better performance than what it displays.

We also plot the relevance of options for discrete control tasks implemented in Miniworld. Option-critic in discrete control tasks doesn't suppress the worse option very quickly. For this reason the difference in performance isn't as drastic as that observed in continuous control tasks. However, there is still a considerable difference between TDEOC and OC. TDEOC always shows a more equitable selection of both options than OC. This may attribute in the slight improvement in performance of TDEOC against OC. However, we believe the major reason for this difference in performance is due to the generation of more interpretable and reusable options. Please refer to Appendix A for implementation details.



((a)) Walker2d-v2

((b)) Ant-v2

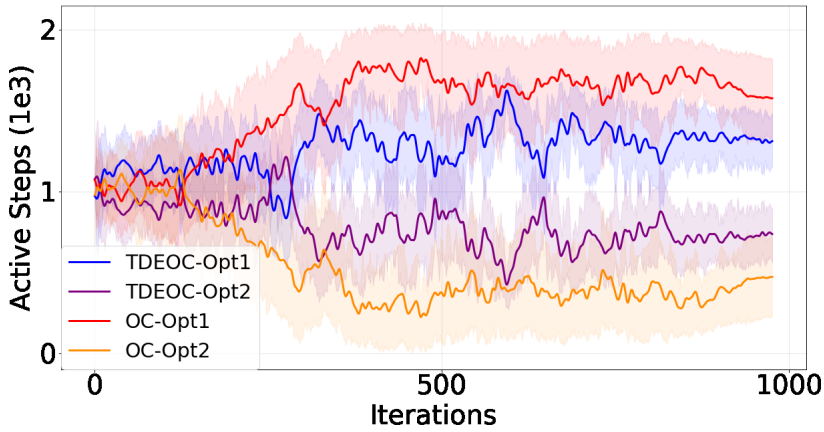
((c)) Hopper-v2

((d)) Humanoid-v2

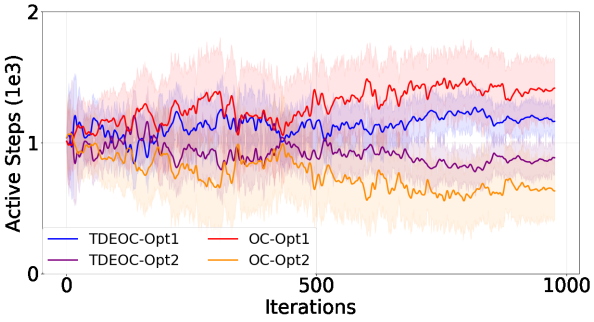
Figure 4.8: **Option activity for standard Mujoco** tasks implemented in Mujoco recorded for ten million steps for Humanoid-v2 while other simulations were run for two million steps (1 Iteration = 2048 steps). Plots are averaged over 20 independent runs.

4.3 Transfer Properties

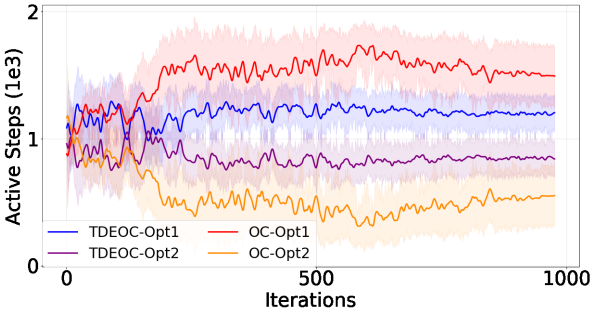
A key advantage of using options in reinforcement learning is to learn generalized skills which can be reused in similar tasks. In section 4.1.1, we studied this property in the tabular four-rooms task. In this section, we further test our objective on certain continuous control tasks where adapting to changes in the task is essential. The benefits of using options are best observed in tasks where hierarchical representation can be exploited. For transfer trials, we compare our algorithm (TDEOC) against Option-Critic (OC) and PPO. The three tasks represent different conditions for adaptation and recovering from events or changes in the environment.



((a)) Sidewalk



((b)) TMaze (Discrete)

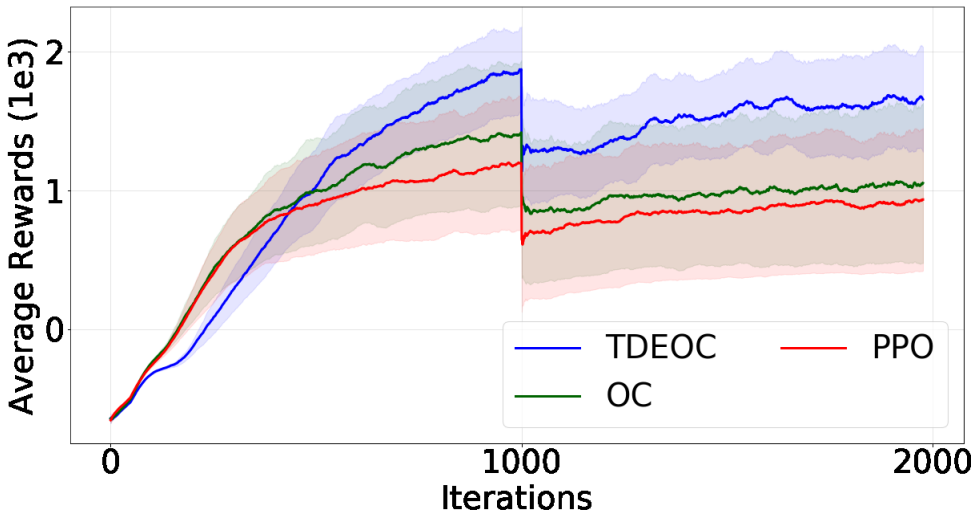


((c)) OneRoom

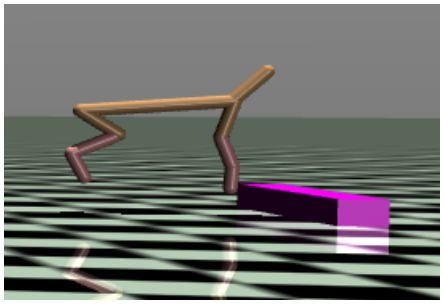
Figure 4.9: **Option activity for standard Miniworld** tasks recorded for two million steps (1 Iteration = 2048 steps). Plots are averaged over 20 independent runs.

4.3.1 HalfCheetah Hurdle

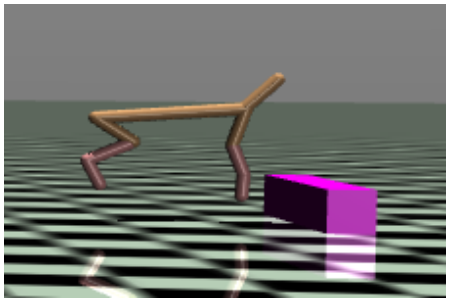
Through this experiment we evaluate the ability of the agent to react to changes in an event further down the trajectory. We reuse the HalfCheetah-v2 simulation from Mujoco. A hurdle of height 0.12 metres is placed 10m away from the agent's starting position. After 1000 iterations, the height of the hurdle is increased by 0.8 metres (to 2 metres). We use the gym-extensions package (Henderson et al. 2017) to add the hurdle and sensors in the environment. The agent's sensor only picks up the presence of the hurdle when it is a metre away. The sensor data indicating distance to the hurdle is incorporated into the observation data and its value indicates the distance between the agent and the hurdle. Not only does the agent need to establish a high velocity running strategy to maximize returns, it should be able to adapt to the



((a)) HalfCheetahHurdle-v0



((b)) Iterations<1000



((c)) Iterations>1000

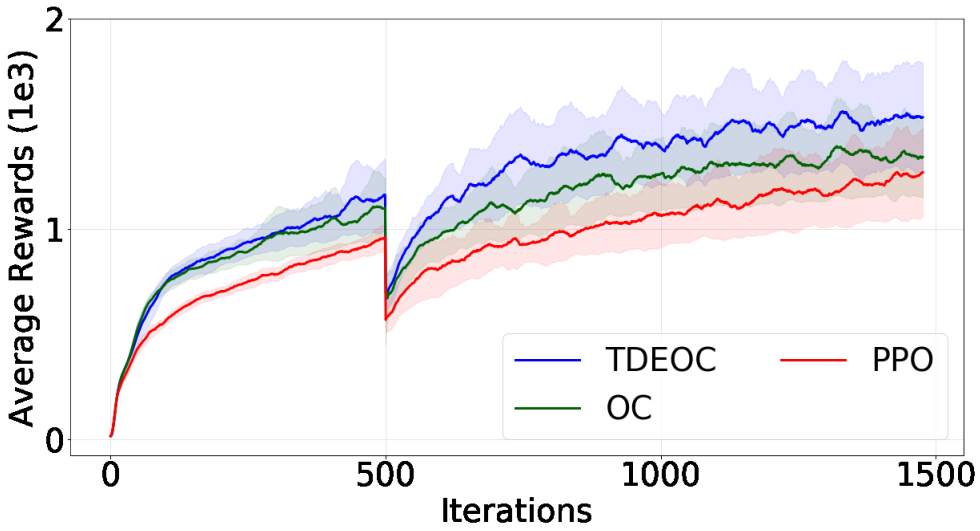
Figure 4.10: **TDEOC results for HalfCheetahHurdle-v0 task** implemented in Mujoco. After a 1000 iterations the height of the hurdle is increased from 1.2m to 2.0 metres. Each line is an average of 20 independent runs (1 Iteration = 2048 steps).

hurdle by quickly learning a leaping technique without flipping over before crossing the wall. Although the agent can learn to move forward on its back, flipping over before crossing the hurdle would make leaping over the hurdle exceedingly difficult. The initial performance of TDEOC, OC and PPO are consistent with experiments on standard Mujoco tasks evaluated in section 4.1.2. TDEOC attains a much higher velocity than OC and PPO prior to changing the height of the hurdle. Upon increasing the height of the hurdle we observe TDEOC recovers much better from the difference in the drop at the 100th iteration. Not only does TDEOC recover better, it also manages to keep improving while OC fails to show much improvement after the change in the hurdle's height. Option-critic however, performs significantly better than PPO both

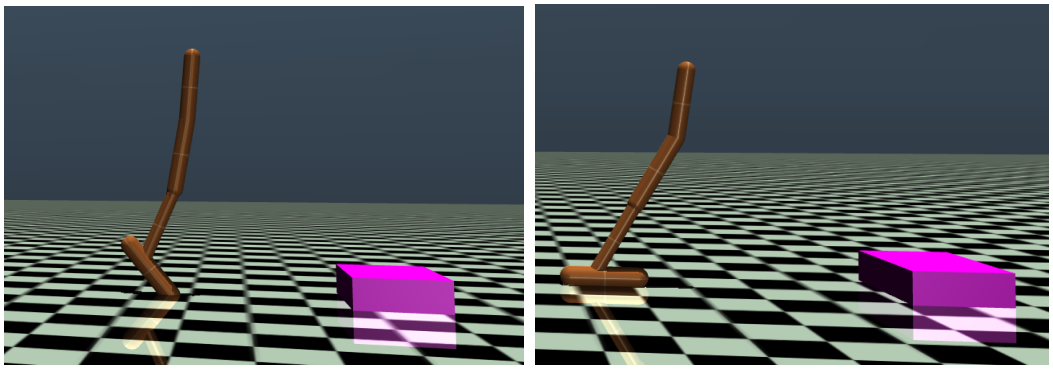
in learning the task of leaping over a hurdle and its adaptation upon the change in height. A performance achieved by TDEOC is especially difficult to achieve when the agent has learned to perform the initial task with significant speed as achieved by TDEOC. At higher speeds it becomes very easy for the agent to flip over or trip upon encountering the hurdle and these instances can affect performance significantly. TDEOC is able to respond to such a change better by terminating around the hurdle region and stabilizing the jump. TDEOC hence retains all the advantageous properties of learning options including the ability to adapt to changes and events observed along the trajectory.

4.3.2 Hopper Ice Wall

We consider the Hopper-v2 environment from Mujoco for this experiment as balance is one of the most vital elements in the task. Our experiments presented in section 4.1.2 show that in the firsts few iterations TDEOC only shows a slight improvement in performance over OC. We now study how TDEOC and OC which, while exhibiting comparable performance before the change, recover once a change is induced. We again use the gym extensions (Henderson et al. 2017) to add a friction-less block with dimensions (0.25, 0.4, 0.12) corresponding to its length, width and height respectively, 2.8 metres away from the agent's starting position. The agent is required to learn how to jump over the block, slide over it without losing balance and resume hopping after. After 500 iterations, the block is moved 0.5 metres away from the agent's starting position. Such a change requires the agent to re-evaluate its technique for jumping onto the block and not losing balance when making contact with it. For an environment such as this, even a small change in its location affects the outcome significantly. The sensor again only notifies the agent of the upcoming object when it is within one metre from it. From figure 4.11(a), we can observe that OC and TDEOC indeed do demonstrate comparable performance for the firsts 500 iterations. Both TDEOC and



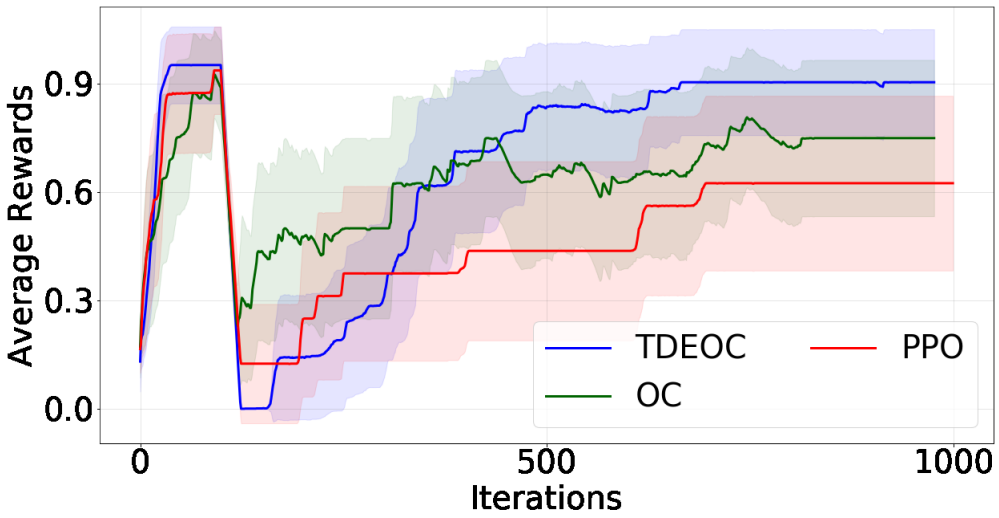
((a)) HopperIceWall-v0



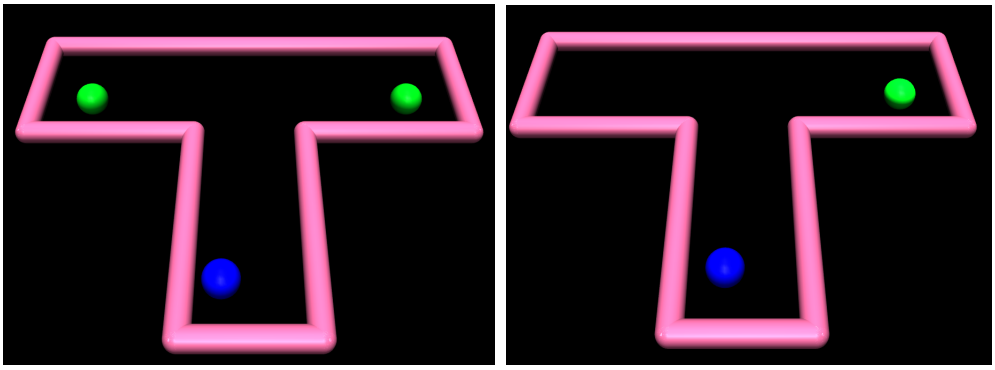
((b)) Iterations<500 ((c)) Iterations>500

Figure 4.11: **TDEOC results for HopperIceWall-v0 task** implemented in Mujoco. After a 500 iterations the block is moved 0.5 metres away from the agent's starting position. Each line is an average of 20 independent runs (1 Iteration = 2048 steps).

OC significantly outperform PPO. After the change is induced in the environment, TDEOC learns to stabilize itself better than OC as well as PPO. TDEOC's ability to recover better than OC despite showing similar performance before validates that TDEOC's learning strategy is more generalized with useful and specialized options which aid easier transfer. OC however, performs significantly better than PPO. Another observation retrieved from Fig 4.11(a) is that TDEOC besides being able to recover faster than OC and PPO, also manages to keep improving.



((a)) TMaze(Continuous)



((b)) Iterations<100 ((c)) Iterations>100

Figure 4.12: **TDEOC results for TMaze-v0 transfer task** implemented in Mujoco. After a 100 iterations the most frequent goal is removed. Each line is an average of 20 independent runs (1 Iteration = 2048 steps).

4.3.3 TMaze Continuous

Next we consider a continuous control task with a sparse reward. We have evaluated the performance of T-Maze from Miniworld in section **??** with discrete action space and a visual input. We now study the transfer abilities of a similar task with continuous control space (Khetarpal et al. 2020) shown in Fig 4.12. There are two goals located at both ends of the hallway each producing a reward of +1. After 100 iterations, the goal most visited is removed forcing the agent to seek the other goal. The agent's actions are indicated by directional forces applied on the ball. The reward annealing step used by Miniworld experiments is also omitted. The agent receives a

reward of +1 irrespective of the number of steps taken to reach it. Due to the sparse reward problem, we do not augment the reward function with $\mathcal{R}_{bonus}$ but still compute it to estimate the terminations. Using this task, we can evaluate the agent's ability to seek an orthogonal solution than its current strategy quickly. Before the most frequent goal is removed, we observe TDEOC manages to outperform OC and PPO exhibiting better learning as well sample efficiency (Fig 4.12(a)). Upon removing the most frequent goal, OC recovers much significantly faster than TDEOC as well as PPO. However, TDEOC quickly surpasses OC and achieves a much better final performance. We reason the performance with illustrations of how option terminations are localized in section 4.4.

4.4 Interpreting Option Behavior

Hierarchical reinforcement learning algorithms can be used to learn specialized lower-tier policies which can offer better interpretibility and understanding of the agent's learning paradigm. With the emergence of deep learning, efficient interpretibility during learning becomes challenging. Using a set of specialized options can not only help understand each option's contribution towards learning the task but also how they behave with respect to other options. An efficient use of options can help the agent learn a more general set of policies which can improve the agent's ability to adapt to changes or perturbations in the environment as demonstrated in section 4.3. An intuitive representation of specialized options can help develop algorithms which can learn increasingly complex tasks without the same strain on computing resources. Learning intuitive and interpretable options in the context of model-free learning is an ongoing challenge. Note that for all the experiments presented in this thesis, there has been no restriction or assumption imposed on an option's initiation set. The models are also initiated without any prior information related to the task indicating an end-to-end approach in learning options without any expert guidance

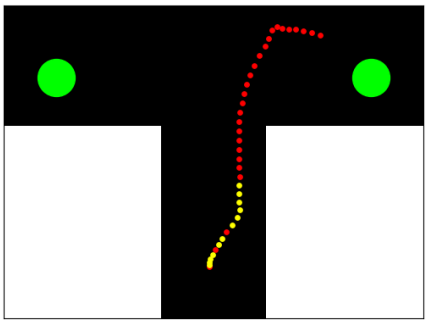
or supervision.

The proposed termination objective (Eq (4.1)) can be interpreted as a function for identifying bottle-neck states where options grow most diverse. In this section, we study how how each option behaves in the context of learning a task using the proposed termination objective. In previous sections we have stressed the importance of identifying and learning certain critical states in the environment. In this section, we visualize each option's spatial orientation thereby highlighting such states and shed light on how such representation proves advantageous to performance.

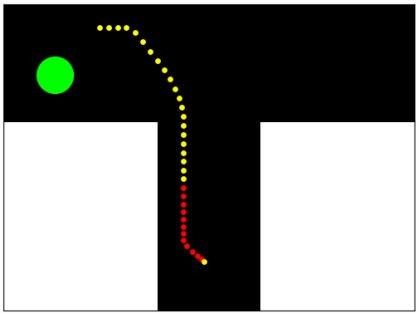
4.4.1 TMaze

In section 4.3.3, we demonstrated the transfer capabilities of TDEOC in the TMaze task (Khetarpal et al. 2020) with continuous control. Empirical analysis shows that TDEOC's final performance exceeds that of OC and PPO both prior the change in the task as well as after the change in the task. We now collaborate the empirical results with visualizations which represent how options are localized on a trajectory. The spatial orientations of options are mainly influenced by the termination condition. We visualize on such run (Fig 4.13). We can observe that terminations are localized in the straight hallway towards the T-junction. This seems to be a very intuitive representation as the choice of navigating to either of the goals is still open in that hallway. This implied the presence of two drastically diverse navigation strategies corresponding to the routes to both the goals from the straight hallway. Once the agent determines which goal it wants to navigate to, a single option is selected to navigate to that goal without terminations. Hence the most diverse options originate at the region where terminations are localized. The trajectories visualized in Fig 4.13 acknowledge this theory with both options clearly distinct both spatially as well as in behavior. Alternatively, another region where options can be most diverse is the T-junction itself. In this scenario, a single option can navigate till the junction without

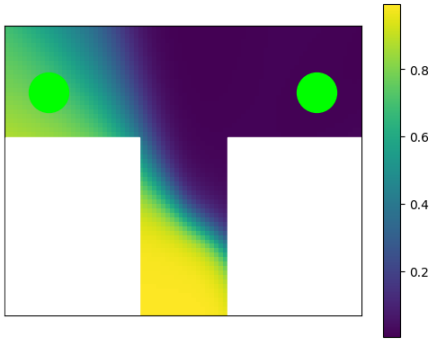
terminating, and the choice of goal to navigate to is taken at the junction where terminations can localize. Another interesting observation is that the terminating region remains practically unchanged upon removing the most frequent goal. This signifies that the strategy adopted by the agent prior to the change was intuitive as well as general enough to retain its orientation even when the task undergoes a change. The termination objective hence helps give rise to intuitive and reusable option behaviors which explain the transfer capabilities of TDEOC.



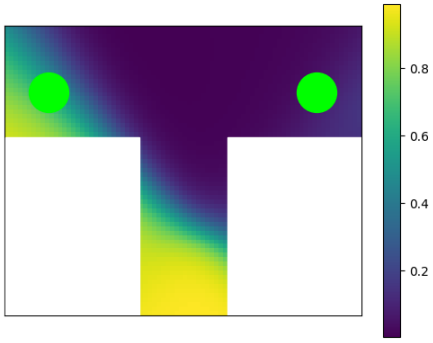
((a)) Trajectory before task change



((b)) Trajectory after most frequent goal is removed



((c)) Terminations for option 1 (β_{o1})



((d)) Terminations for option 2 (β_{o2})

Figure 4.13: **Visualizations on TMaze task using two options** (marked red and yellow respectively in (a) and (b)). Option terminations localize in the vertical hallway where the agent has yet to decide which goal to navigate towards.

4.4.2 OneRoom

We now analyze the options learned in the OneRoom environment from Miniworld. We show in Fig 4.6 that TDEOC achieved a better performance when compared

to OC. We now visualize how options localize on a trajectory. Note that TDEOC doesn't impose any spatial restrictions on option's initiation which is why the spatial arrangement of options may keep changing slightly throughout the learning cycle or accross runs. We visualize one such trajectory (Fig 4.14). One option learns to scan the room by turning on the spot, and upon observing the goal in its visual scope, the second option navigates towards it. This seems to be a very intuitive strategy generated by distinguishing options by behavior. Not only does this retain both options to remain useful, it also exploits them in a way which benefits performance as well interpretibility.

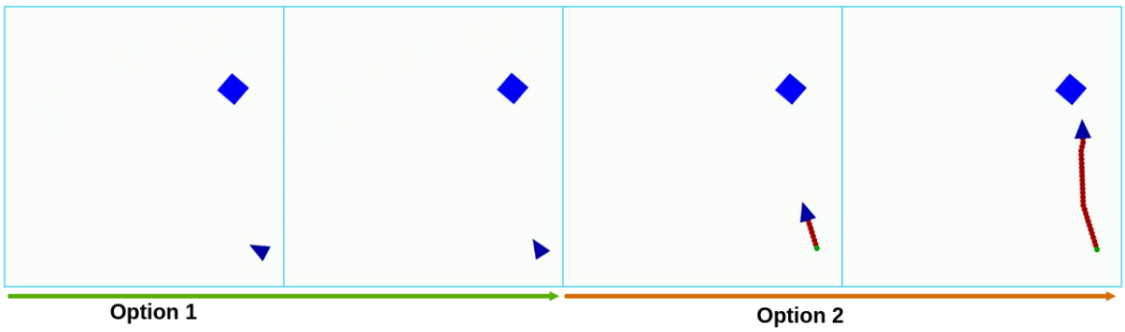


Figure 4.14: **Option trajectories in OneRoom task.** The first options scans the environment for the goal while the other option moves forward towards it.

4.4.3 Hopper-v2

In section 4.1.2, we show that not only does TDEOC demonstrate better performance, it also handles perturbations in the environment better. We evaluate how two options collectively help stabilize the agent and observe the states where such stability is crucial. We consider the Hopper-v2 simulation from Mujoco. From Fig 4.15, we can see that options terminate when the agent is in the air just before descending. Naturally such an instance in the agent's trajectory is extremely vital as even a slight mistake can cause the agent to lose balance and cause the episode to end. TDEOC manages to employ both the options around these states to complement

each other thereby achieving robust control. Hence, TDEOC is capable of achieving better performance while showing much lower variance.

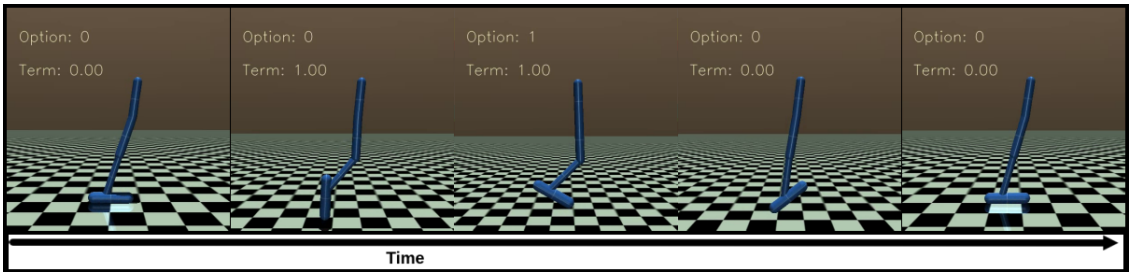
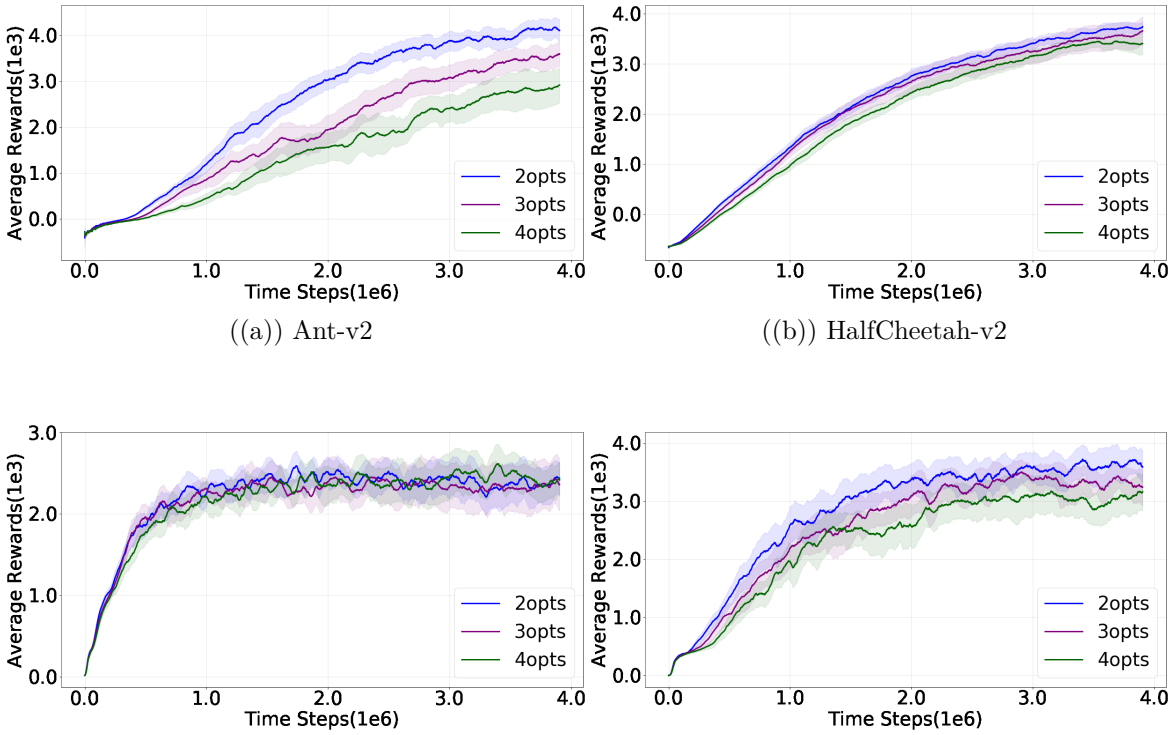


Figure 4.15: **Sample trajectory of the Hopper-v2 task.** Terminations are localized near states where the agent is in the air. Both options collaborate to ensure proper posture and balance prior to descending.

4.5 Effects of Varying the Number of Options

Option-critic framework assumes the number of options to be learnt, to be a hyper-parameter. Generalization of an reinforcement learning algorithm to variable learning parameters adds to its value. Option-critic is capable of learning any number of options specified by the user without any explicit changes to its framework. However, easily scaling to more options without compromising too much on performance and sample complexity is still a challenge. Moreover, the algorithm should be able to specifically use all the options instead of a select few to perform the task. In this section, we show that our approach not only easily generalizes to variable number of options, it is still able to outperform option-critic on respective number of options (Fig. 4.18 and Fig. 4.19). The differences in performance in Fig. 4.16 and Fig. 4.17 is owed to the increased sample complexity while learning more options. This phenomenon is particularly evident in TDEOC results. This is due to the fact that TDEOC encourages all available options to remain relevant which consequently requires more samples to achieve comparable performance benchmark. The effect of

varying options is not as significant in the case of option-critic as it still suffers from degeneration where a single option quickly dominates over the entire task. To ensure reproducible results and fair comparisons, the choice of the hyper-parameter, number of options, for all experiments presented in the paper relied mainly on previously tested experiments (refer Appendix A.1). We reuse the PPOC codebase (Dhariwal et al. 2017; Klissarov et al. 2017) for all our experiments involving non-linear function approximation. PPOC has only been tested on two options (Klissarov et al. 2017) as of yet which is why evaluations using two options are conducted throughout the rest of the thesis. However, our approach easily generalizes to any number of options.



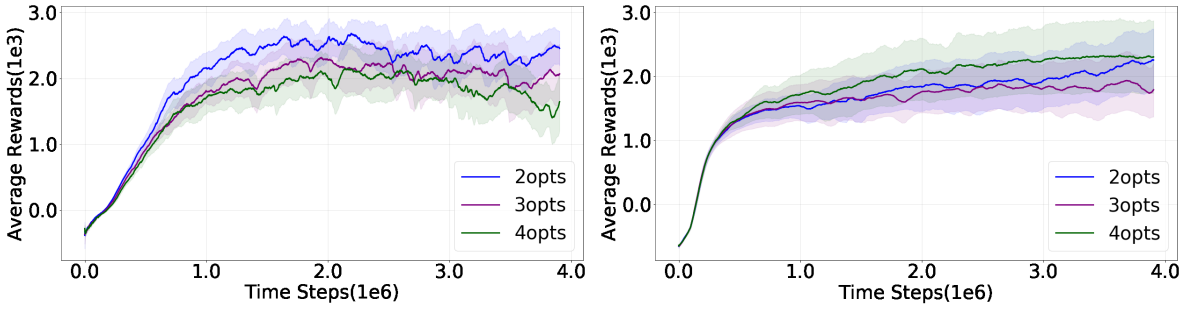
((a)) Ant-v2

((b)) HalfCheetah-v2

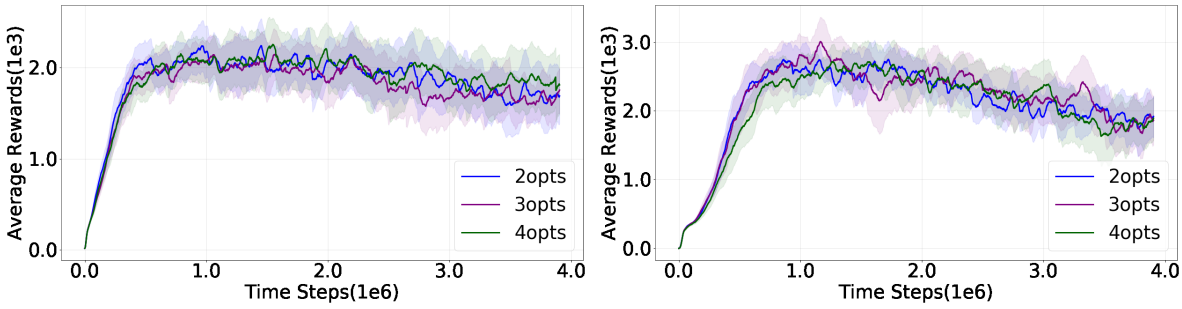
((c)) Hopper-v2

((d)) Walker2d-v2

Figure 4.16: **TDEOC results on four Mujoco tasks with varying number of options.** Sample complexity keeps growing with increasing the number of options. Each line is an average of 20 runs.

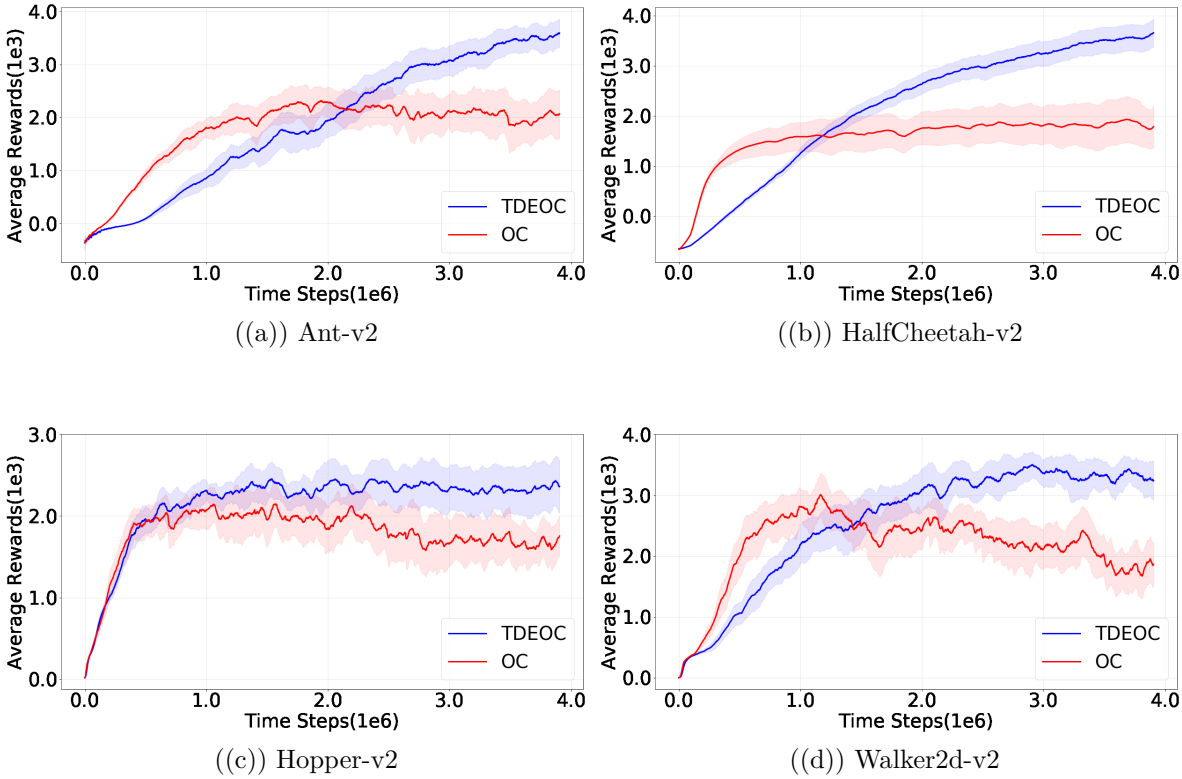


((a)) Ant-v2 ((b)) HalfCheetah-v2



((c)) Hopper-v2 ((d)) Walker2d-v2

Figure 4.17: **Option-Critic results on four Mujoco tasks with varying number of options.** Each line is an average of 20 runs.



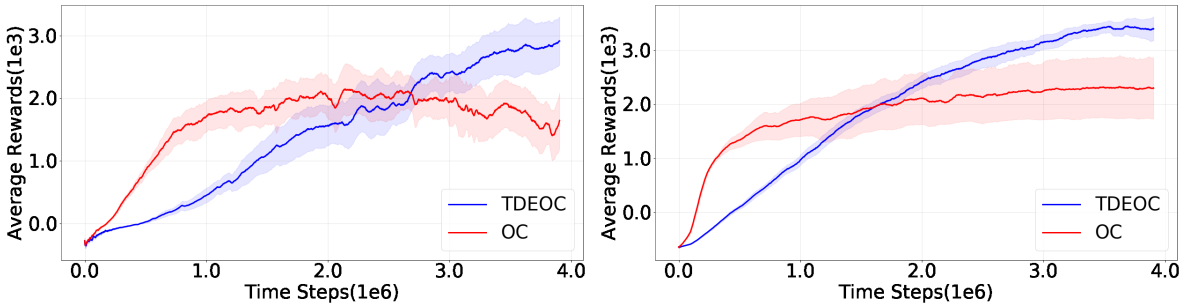
((a)) Ant-v2

((b)) HalfCheetah-v2

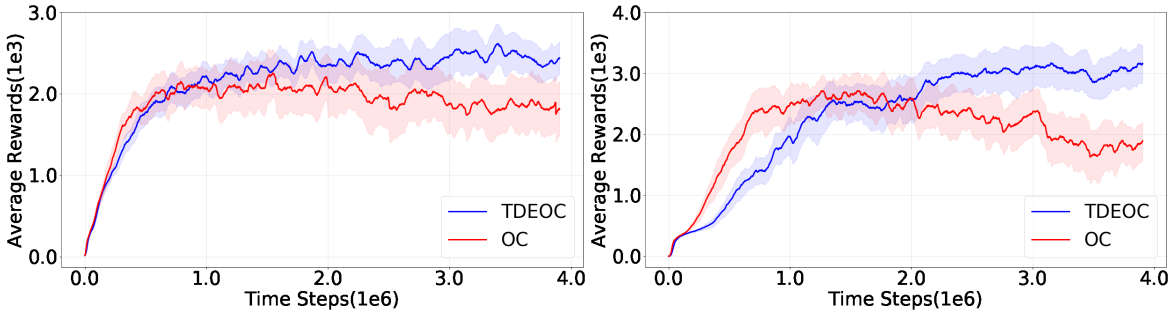
((c)) Hopper-v2

((d)) Walker2d-v2

Figure 4.18: **TDEOC and OC results on four Mujoco tasks with three options.** Each line is an average of 20 runs.



((a)) Ant-v2 ((b)) HalfCheetah-v2



((c)) Hopper-v2 ((d)) Walker2d-v2

Figure 4.19: **TDEOC and OC results on four Mujoco tasks with four options.** Each line is an average of 20 runs.

5

Conclusion

In this thesis we highlight the importance of diversity in learning options. Using concepts of information theory, we derived a information-theoretic pseudo-reward which measures how diverse two options are based on their behavior at any given state. This pseudo-reward coupled with the task reward encouraged options to grow diverse while simultaneously being able to learn the task. We present Diversity Enriched Option-Critic (DEOC), which encourages options to adopt diverse strategies to improve an agent's exploration and performance solely using reward augmentation.

We then propose a novel termination objective which focuses on identifying states in an environment where options can be most diverse. These critical states represent bottle-neck states in the environment which are often the most crucial and sensitive states in the environment. The proposed termination objective encourages both options to be fairly and adequately explored around these states while increasing diversity. Since the proposed termination objective doesn't rely on validating the best available option, both options remain relevant and useful to the task. We present Termination-DEOC (TDEOC), a diversity-motivated learning algorithm combining the intrinsic reward augmentation of DEOC as well as the new termination objective. We empirically demonstrate that not only does TDEOC exhibits improved exploration as well as overall performance on a wide array of tasks, it also handles environment perturbations better by stabilizing the sensitive and vulnerable regions of the envi-

ronment using a set of diverse yet useful options. This is also the reason why TDEOC results show lower variance when compared to OC. Not only does TDEOC outperform OC on standard control tasks, it also retains all the useful properties presented by OC such as transfer capabilities and interpretibility. TDEOC has consistently surpassed the results achieved by OC on all the transfer tasks performed. Visualizations of option behavior for TDEOC show a very intuitive learning representations of option behaviors. Its ability to identify the bottleneck states also allows TDEOC to quickly identify and learn events and changes in the environment throughout its trajectory. This property is quite useful in scaling up learning to longer learning horizons. It is interesting to note that TDEOC manages to achieve all the results without any restriction on option's initiation, without any explicit prior knowledge nor any expert supervision.

Despite all the advantages that TDEOC offers, its biggest shortcoming is sample efficiency. Since TDEOC explores all available options around states sparking terminations, sample complexity would keep increasing with an increase in the number of options. An elegant solution to this would be to restrict the number of option choices at a given state using an attention-based approach such as interest functions. Such a strategy can easily help the algorithm learn longer horizon tasks with multiple events uniquely over the agent's trajectory while preserving its ability to reuse specialized options upon recurrence on any event. The issue of sample complexity can also be somewhat mitigated using an off-policy learning architecture such as Soft Actor-Critic instead of an on-policy algorithm such as PPOC.

Bibliography

Bacon, Pierre-Luc (2013). "On the bottleneck concept for options discovery". PhD thesis. Masters thesis, McGill University, 2013.

— (2018). "Temporal Representation Learning". PhD thesis. PhD thesis, McGill University, 2018.

Bacon, Pierre-Luc, Jean Harb, and Doina Precup (2017). "The option-critic architecture". In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.

Bagaria, Akhil and George Konidaris (2020). "Option Discovery using Deep Skill Chaining". In: *International Conference on Learning Representations*. 2020. https://openreview.net/forum?id=B1gqipNYwH.

Bellemare, Marc et al. (2016). "Unifying count-based exploration and intrinsic motivation". In: *Advances in Neural Information Processing Systems*. 2016, 1471–1479.

Bellman, Richard (Nov. 1954). "The theory of dynamic programming". In: *Bull. Amer. Math. Soc.* 60.6, 503–515. https://projecteuclid.org:443/euclid.bams/1183519147.

Chevalier-Boisvert, Maxime (2018). *gym-miniworld environment for OpenAI Gym*. https://github.com/maximecb/gym-miniworld. 2018.

Deci, Edward L and Richard M Ryan (2010). "Intrinsic motivation". In: *The corsini encyclopedia of psychology*, 1–2.

Dhariwal, Prafulla et al. (2017). *OpenAI Baselines*. https://github.com/openai/baselines. 2017.

Dietterich, Thomas G (1998). "The MAXQ Method for Hierarchical Reinforcement Learning." In: *ICML*. Vol. 98. Citeseer. 1998, 118–126.

Eysenbach, Benjamin et al. (2018). "Diversity is All You Need: Learning Skills without a Reward Function". In: *CoRR* abs/1802.06070. arXiv: 1802.06070. http://arxiv.org/abs/1802.06070.

Gregor, Karol, Danilo Jimenez Rezende, and Daan Wierstra (2016). "Variational intrinsic control". In: *arXiv preprint arXiv:1611.07507*.

Haarnoja, Tuomas et al. (2017). "Reinforcement learning with deep energy-based policies". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, 1352–1361.

Haarnoja, Tuomas et al. (2018). "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: *arXiv preprint arXiv:1801.01290*.

Harackiewicz, Judith M and Andrew J Elliot (1993). "Achievement goals and intrinsic motivation." In: *Journal of personality and social psychology* 65.5, 904.

Harb, Jean et al. (2018). "When waiting is not an option: Learning options with a deliberation cost". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

Harutyunyan, Anna et al. (2019). "The Termination Critic". In:

Hauskrecht, M et al. (1998). "Hierarchical solutions of MDPs using macro-actions". In: *Proc. UAI*. Vol. 98. 1998.

Henderson, P. et al. (2017). "Benchmark Environments for Multitask Learning in Continuous Domains". In: *ICML Lifelong Learning: A Reinforcement Learning Approach Workshop*.

Iba, Glenn A (1989). "A heuristic approach to the discovery of macro-operators". In: *Machine Learning* 3.4, 285–317.

Khetarpal, Khimya et al. (2020). "Options of Interest: Temporal Abstraction with Interest Functions". In: *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20)*.

Klissarov, Martin et al. (2017). "Learnings Options End-to-End for Continuous Action Tasks". In: *ArXiv* abs/1712.00004.

Korf, Richard E (1983). *Learning to solve problems by searching for macro-operators*. Tech. rep. CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE, 1983.

Levy, Andrew, Robert Platt, and Kate Saenko (2018). "Hierarchical reinforcement learning with hindsight". In:

Machado, Marlos C, Marc G Bellemare, and Michael Bowling (2018). "Count-based exploration with the successor representation". In: *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI 2020)*.

Mann, Timothy, Daniel Mankowitz, and Shie Mannor (22–24 Jun 2014). "Time-Regularized Interrupting Options (TRIO)". In: *Proceedings of the 31st International Conference on Machine Learning*. Ed. by Eric P. Xing and Tony Jebara. Vol. 32. Proceedings of Machine Learning Research 2. Bejing, China: PMLR, 22–24 Jun 2014, 1350–1358. http://proceedings.mlr.press/v32/mannb14.html.

McGovern, Amy and Andrew G. Barto (2001). "Automatic Discovery of Subgoals in Reinforcement Learning Using Diverse Density". In: *Proceedings of the Eighteenth International Conference on Machine Learning*. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, 361–368. ISBN: 1558607781.

Mnih, Volodymyr et al. (2015). "Human-level control through deep reinforcement learning". In: *Nature* 518.7540, 529–533.

Mnih, Volodymyr et al. (2016). "Asynchronous Methods for Deep Reinforcement Learning". In: *ICML*. 2016.

Ng, Andrew Y, Daishi Harada, and Stuart Russell (1999). "Policy invariance under reward transformations: Theory and application to reward shaping". In: *ICML*. Vol. 99. 1999, 278–287.

Ostrovski, Georg et al. (2017). "Count-based exploration with neural density models". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, 2721–2730.

Parr, Ronald and Stuart J Russell (1998). "Reinforcement learning with hierarchies of machines". In: *Advances in neural information processing systems*. 1998, 1043–1049.

Precup, Doina (2000). *Temporal Abstraction in Reinforcement Learning*. 2000.

Sacerdoti, Earl D (1974). "Planning in a hierarchy of abstraction spaces". In: *Artificial intelligence* 5.2, 115–135.

Sansone, Carol and Judith M Harackiewicz (2000). *Intrinsic and extrinsic motivation: The search for optimal motivation and performance*. Elsevier, 2000.

Schulman, John et al. (2015). "Trust Region Policy Optimization". In: *ICML*. 2015.

Schulman, John et al. (2017). "Proximal Policy Optimization Algorithms". In: *ArXiv* abs/1707.06347.

Shannon, Claude E (1948). "A mathematical theory of communication". In: *Bell system technical journal* 27.3, 379–423.

Singh, Satinder P (1992). "Scaling reinforcement learning algorithms by learning variable temporal resolution models". In: *Proceedings of the Ninth International Machine Learning Conference*. 1992, 406–415.

Singh, Satinder et al. (2010). "Intrinsically motivated reinforcement learning: An evolutionary perspective". In: *IEEE Transactions on Autonomous Mental Development* 2.2, 70–82.

Stolle, Martin and Doina Precup (2002). "Learning Options in Reinforcement Learning". In: *Abstraction, Reformulation, and Approximation*. Ed. by Sven Koenig and

Robert C. Holte. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, 212–223. ISBN: 978-3-540-45622-3.

Sutton, Richard S (1985). "Temporal Credit Assignment in Reinforcement Learning." In:

Sutton, Richard S and Andrew G Barto (2018). *Reinforcement learning: An introduction*. MIT press, 2018.

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

Tang, Haoran et al. (2017). "OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel.# exploration: A study of count-based exploration for deep reinforcement learning". In: *Advances in neural information processing systems* 30, 2753–2762.

Todorov, Emanuel, Tom Erez, and Yuval Tassa (2012). "Mujoco: A physics engine for model-based control". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, 5026–5033.

Vezhnevets, Alexander Sasha et al. (2017). "FeUdal Networks for Hierarchical Reinforcement Learning". In: *ICML*. 2017.

Watkins, Christopher JCH and Peter Dayan (1989). "Q-learning". In: *Machine learning* 8.3-4, 279–292.

Williams, Ronald J and Jing Peng (1991). "Function optimization using connectionist reinforcement learning algorithms". In: *Connection Science* 3.3, 241–268.

Zheng, Zeyu, Junhyuk Oh, and Satinder Singh (2018). "On learning intrinsic rewards for policy gradient methods". In: *Advances in Neural Information Processing Systems*. 2018, 4644–4654.

A

Implementation Details

A.1 Implementation Details

A.1.1 Choice of Underlying algorithm

Our proposed termination objective relies on diversity estimates drawn from a well updated policy in order to identify the states where options tend to grow most diverse. This property is best observed in on-policy algorithms where samples from the same policy is used for policy evaluation and policy improvement. PPO is the state-of-the-art on-policy algorithm which is why it was used as the underlying algorithm. Although off-policy algorithms such as TD3 or SAC may achieve superior performance, these methods draw sample from a large buffer (mostly of capacity one million samples) which may even contain samples from the initial random policy, which dilutes TDEOC's ability to target most diverse states. Also, unlike PPO, TD3 and SAC have only been implemented for continuous control tasks while PPO has also been tested on discrete control tasks such as Miniworld (Chevalier-Boisvert 2018). Another significant reason for choosing PPO is because option-critic had already been implemented using PPO updates (PPOC) (Dhariwal et al. 2017; Klissarov et al. 2017) and had been tested extensively making our experiments standard, fair and reproducible.

A.1.2 Tabular Case

For our four-rooms experiments, we reuse the implementations by bacon2017option. Augmenting the reward with $\mathcal{R}_{bonus}$ for tasks with very sparse rewards can cause the agent to prioritize diversifying options over learning the task. To mitigate this, we avoid the reward augmentation step in all our sparse reward tasks including the four-rooms task. Instead of standardizing the diversity values, use update a moving sum of all values observed in the current run and center the diversity around the moving mean instead. For more than three options, the diversity is computed by sampling six pairs of options and averaging the respective cross entropy. The sole reason of this is to avoid computing the mean of all samples at every step. Each algorithm is averaged over 300 runs. The code has been attached in the submission folder.

Hyper-parameter	TDEOC	OC
Termination lr	5e-2	1e-1
Intra-Option lr	1e-2	1e-2
Critic lr	5e-1	5e-1
Action Critic lr	5e-1	5e-1
Discount	0.99	0.99
Max Steps	1000	1000
No of Options	4	4
Temperature	1e-3	1e-3

Table A.1: Hyper-parameters for Tabular Four-rooms task

A.1.3 Non-Linear Function Approximation Case

We provide the implementation details as well as the hyper-parameters for all our non-linear function approximation cases. The hyper-parameters for PPO and OC are consistent with those suggested in baselines and Klissarov2017LearningsOE respectively. We use two critics for our algorithms (DEOC and TDEOC) as used by bacon2017option in their tabular case implementation. All our plots are averaged over 20 independent runs with the error bounds representing 0.5 of the standard deviation.

A.1.3.1

For standard Mujoco tasks, we incorporate our algorithm within the PPOC code (Klissarov et al. 2017). The pseudo reward bonus is scaled down depending on the task with the intention of prioritizing task reward. The diversity term is calculated using cross entropy, as stated in the Eq. (3.1). We compute the softmax of the action distribution before computing the cross entropy to ensure $\mathcal{R}_{bonus}$ remains positive. As mentioned earlier, for the TMaze(continuous) task, we avoid augmenting the reward with the diversity term. As for the HalfCheetahWall-v0 and HopperIce-v0, we reuse the resources provided by (Henderson et al. 2017). The obstacle is observed in the agent's state space only when the agent is within one metre of the obstacle. The learning rate for TDEOC is slightly slower than OC to allow the algorithm sufficient time to learn the bottleneck states. We average runs over 20 sequential seeds starting from seed 10 for our results.

Hyper-parameter	Value
Termination lr	5e-7
Termination lr TMaze(Continuous)	5e-8
Timesteps per batch	2048
Optim epochs	10
Clip param	0.2
Entropy coefficient	0.0
Gamma	0.99
Lambda	0.95
Lr schedule	constant

Table A.2: Common hyper-parameters across all continuous control tasks

A.1.3.2

For Miniworld tasks, we use the code provided by baselines (Dhariwal et al. 2017) for Atari environments and implement the PPOC networks consistent with (Klissarov et al. 2017) within. The objective of the Sidewalk task is for the agent to navigate to a goal object placed at the end of the street. The episode terminates and the

Environment	TDEOC	DEOC	OC
Ant-v2	1e-4	3e-4	3e-4
HalfCheetah-v2	1e-4	3e-4	3e-4
Hopper-v2	1e-4	3e-4	3e-4
Walker2d-v2	1e-4	3e-4	3e-4
Humanoid-v2	3.33e-5	1e-4	1e-4
HalfCheetahWall-v0	1e-4	nan	1e-4
HopperIceWall-v0	1e-4	nan	1e-4
TMaze (Continuous)	3e-5	nan	1e-4

Table A.3: Learning rates for various continuous control tasks

Environment	Trade-off
Ant-v2	0.2
HalfCheetah-v2	0.7
Hopper-v2	0.2
Walker2d-v2	0.2
HalfCheetahWall-v0	0.6
HopperIceWall-v0	0.4
TMaze (Continuous)	0.0

Table A.4: Trade-off value for various control tasks

environment resets if the agent strays away from the path onto the street. The OneRoom task can be perceived as a simpler version of the Sidewalk task, solely consisting of navigating to the goal object placed randomly anywhere in the room. As for the TMaze environment, the objective is to navigate to a goal object placed randomly at either ends of the horizontal bottom hallway. Due to discrete action space, diversity is computed by taking the softmax of the logits of the policy network. Rest of the implementation is consistent with the continuous control case discussed above. Due to the sparse reward situation, we again do not augment the reward with the diversity term. Like in the continuous control case, we average runs over 20 sequential seeds starting from seed 10 for our results. The hyper parameters used are given below:

Hyper-parameter	Value
Termination lr	5e-7
Timesteps per batch	2048
Optim epochs	4
Entropy coefficient	0.1
Clip Param	0.2
Gamma	0.99
Lambda	0.95
Lr schedule	linear

Table A.5: Common hyper-parameters across all Miniworld tasks

Environment	TDEOC	OC
MiniWorld-OneRoom-v0	1e-4	3e-4
MiniWorld-Sidewalk-v0	1e-4	3e-4
MiniWorld-TMaze-v0	1e-4	3e-4

Table A.6: Learning rates for various Miniworld tasks