Options of Interest: Temporal Abstraction with Interest Functions

Khimya Khetarpal, Martin Klissarov, Maxime Chevalier-Boisvert, Pierre-Luc Bacon, Doina Precup

1McGill University, 2Mila, 3Université de Montréal, 4Stanford University, 5Google DeepMind

khimya.khetarpal@mail.mcgill.ca, martin.klissarov@mail.mcgill.ca, maxime.chevalier-boisvert@mila.quebec, plbacon@cs.stanford.edu, dprecup@cs.mcgill.ca

Abstract

Temporal abstraction refers to the ability of an agent to use behaviours of controllers which act for a limited, variable amount of time. The options framework describes such behaviours as consisting of a subset of states in which they can initiate, an internal policy and a stochastic termination condition. However, much of the subsequent work on option discovery has ignored the initiation set, because of difficulty in learning it from data. We provide a generalization of initiation sets suitable for general function approximation, by defining an interest function associated with an option. We derive a gradient-based learning algorithm for interest functions, leading to a new interest-option-critic architecture. We investigate how interest functions can be leveraged to learn interpretable and reusable temporal abstractions. We demonstrate the efficacy of the proposed approach through quantitative and qualitative results, in both discrete and continuous environments.

1 Introduction

Humans have a remarkable ability to acquire skills, and knowing when to apply each skill plays an important role in their ability to quickly solve new tasks. In this work, we tackle the problem of learning such skills in reinforcement learning (RL). AI agents which aim to achieve goals are faced with two difficulties in large problems: the depth of the lookahead needed to obtain a good solution, and the breadth generated by having many choices. The first problem is often solved by providing shortcuts that skip over multiple time steps, for example, by using macro-actions [13]. The second problem is handled by restricting the agent’s attention at each step to a reasonable number of possible choices. Temporal abstraction methods aim to solve the first problem, and a lot of recent literature has been devoted to this topic [16, 23, 18, 21, 3, 20]. We focus specifically on the second problem: learning how to reduce the number of choices considered by an RL agent.

In classical planning, the early work on STRIPS [8] used preconditions that had to be satisfied before applying a certain action. Similar ideas can also be found in later work on macro-operators [17] or the Schema model [7]. In RL, the framework of options [36] uses a similar concept, initiation sets, which limit the availability of options (i.e. temporally extended actions) in order to deal with the possibly high cost of choosing among many options. Moreover, initiation sets can also lead to options that are more localized [15], which can be beneficial in transfer learning. For example, in continual learning [32], specialization is key to both scaling up learning in large environments, as well as to “protecting” knowledge that has already been learned from forgetting due to new updates.

The option-critic architecture [3] is a gradient-based approach for learning options in order to optimize the usual long-term return obtained by an RL agent from the environment. However, the
Figure 1: **Interest functions and the branching factor.** During the initial stages of the learning process, allowing fewer options helps improve learning speed, whereas in later stages, good solutions can still be obtained with a reasonable number of choices at each decision point.

The notion of initiation sets originally introduced in [36] was omitted from [3] due to the difficulty of learning sets with gradient-based methods. We propose a generalization of initiation sets to **interest functions** [35, 40]. We build from the fact that a set can be represented through its membership function. Interest functions are a generalization of membership functions which allows smooth parameterization. Without this extension, determining suitable initiation sets would necessitate a non-differentiable, search-based approach.

**Key Contributions:** We generalize initiation sets for options to **interest functions**, which are differentiable, and hence easier to learn. We derive a gradient-based learning algorithm capable of learning all components of options end-to-end. The resulting interest-option-critic architecture generates options that are useful in a single task, interpretable and reusable in multi-task learning.

## 2 Preliminaries

**Markov Decision Processes (MDPs).** A finite, discrete-time MDP [31, 34] is a tuple \( \langle S, A, r, P, \gamma \rangle \), where \( S \) is the set of states, \( A \) is the set of actions, \( r : S \times A \to \mathbb{R} \) is the reward function, \( P : S \times A \to \text{Dist}(S) \) is the environment transition probability function, and \( \gamma \in [0, 1) \) is the discount factor. At each time step, the learning agent perceives a state \( S_t \in S \), takes an action \( A_t \in A \) drawn from a policy \( \pi : S \times A \to [0, 1] \), and with probability \( P(S_{t+1}|S_t, A_t) \) enters next state \( S_{t+1} \), receiving a numerical reward \( R_{t+1} = r(S_t, A_t) \) from the environment. The value function of policy \( \pi \) is defined as: \( V_{\pi}(s) = \mathbb{E}_{\pi}[\sum_{t=0}^{\infty} \gamma^t R_{t+1}|S_0 = s] \) and its action-value function is: \( Q_{\pi}(s, a) = \mathbb{E}_{\pi}[\sum_{t=0}^{\infty} \gamma^t R_{t+1}|S_0 = s, A_0 = a] \).

**The Options framework.** A Markovian option [36] \( \omega \in \Omega \) is composed of an **intra-option policy** \( \pi_\omega \), a termination condition \( \beta_\omega : S \to [0, 1] \), where \( \beta_\omega(s) \) is the probability of terminating the option upon entering state \( s \), and an initiation set \( I_\omega \subseteq S \). In the **call-and-return** option execution model, when an agent is in state \( s \), it first examines the options that are available, i.e., for which \( s \in I_\omega \). Let \( \Omega(s) \) denote this set of available options. The agent then chooses \( \omega \in \Omega(s) \) according to the policy over options \( \pi_\Omega(s) \), follows the internal policy of \( \omega, \pi_\omega \), until it terminates according to \( \beta_\omega \), at which point this process is repeated. Note that \( \Omega \) is the union of all sets \( \Omega(s) \), \( \forall s \). The option-value function of \( \omega \in \Omega(s) \) is defined as:

\[
Q_\Omega(s, \omega) = \sum_a \pi_\omega(a|s)Q_U(s, \omega, a),
\]

where \( Q_U : S \times \Omega \times A \to \mathbb{R} \) is the value of executing primitive action \( a \) in the context of state-option pair \((s, \omega)\):

\[
Q_U(s, \omega, a) = r(s, a) + \gamma \sum_{s'} P(s'|s, a) \left( (1 - \beta_\omega(s'))Q_\Omega(s', \omega) + \beta_\omega(s') \max_{\omega' \in \Omega(s')} Q_\Omega(s', \omega') \right)
\]

Note that if an option cannot initiate in a state \( s \), its value is considered undefined.
3 Interest-Option-Critic

In [36], the focus is on discrete environments, and the notion of initiation set provides a direct analog of preconditions from classical planning. In large problems, options would be applicable in parts of the state space described by certain features. For example, an option of the form stop if the traffic light is red would only need to be considered in states where a traffic light is detected. Let $I_ω : S \to \{0, 1\}$ be the indicator function corresponding to set $I_ω$: $I_ω(s) = 1$ iff $s \in I_ω$ and 0 otherwise.

An interest function $I_ω : S \times Ω \to \mathbb{R}^+$ generalizes the set indicator function, with $I_ω(s) > 0$ iff $ω$ can be initiated in $s$. A bigger value of $I_ω(s)$ means the interest in executing $ω$ in $s$ is larger. Note that, depending on how $I_ω$ is parameterized, one could view the interest as a prior on the likelihood of executing $ω$ in $s$. However, we will not use this perspective here, because our goal is to learn $I_ω$.

So, instead, we will choose a parameterized form for $I_ω$, which is differentiable, in order to leverage the power of gradients in the learning process.

The value of $I_ω$ modulates the probability of option $ω$ being sampled in state $s$ by a policy over options $π_Ω$, resulting in an interest policy over option defined as:

$$\pi_{I_ω}(ω|s) \propto I_ω(s)π_Ω(ω|s)$$

Note that this specializes immediately to usual initiation sets (where the interest is the indicator function).

We will now describe an approach to learning options which includes interest functions. We propose a policy gradient algorithm, in the style of option-critic [3], based on the following result:

**Theorem 1.** Given a set of Markov options with differentiable interest functions $I_{ω, z}$, where $z$ is the parameter vector; the gradient of the expected discounted return with respect to $z$ at $(s, ω)$ is:

$$\sum_{s', ω'} \tilde{μ}_{Ω}(s', ω'|s, ω)β_{ω}(s') \frac{∂π_{I_{ω, z}}(ω'|s')}{∂z}Q_{Ω}(s', ω')$$

where $\tilde{μ}_{Ω}(s', ω'|s, ω)$ is the discounted weighting of the state-option pairs along trajectories starting from $(s, ω)$ sampled from the distribution determined by $π_{I_{ω, z}}$, $β_{ω}$ is the termination function and $Q_{Ω}$ is the value function over options corresponding to $π_{I_{ω, z}}$.

The proof is in Appendix A.2.1. We derive policy gradients for intra-option policies and termination functions as in option-critic [3] (see A.2.2 & A.2.3), with the difference that the discounted weighting of state-option pairs is now according to the new option sampling distribution determined by $π_{I_{ω, z}}(s)$.

This is natural, as the introduction of the interest function should only impact the choice of option in each state. Pseudo-code of the interest-option-critic (IOC) algorithm using intra-option Q-learning is shown in Algorithm 1.

Intuitively, the gradient update to $z$ can be interpreted as increasing the interest in an option which terminates in states with good value. It links initiation and termination, which is natural. It is to be noted that the proposed gradient works at the level of the augmented chain; and not at the SMDP level. Implementing policy gradient at the SMDP level for the policy over options would entail performing gradient updates only upon termination, whereas using the augmented chain allows for updates throughout.

4 Illustration

In order to elucidate the way in which interest functions can help regulate the complexity of learning how to make decisions, we provide a small illustration of the idea. Consider a point mass agent in a continuous 2-D maze, which starts in a uniformly random position and must reach a fixed goal state. Consider a scalar threshold $k \in [0, 1]$, so that at any choice point, only options whose interest is at least $k$ can be initiated by the interest policy over options $π_{I_ω}(ω|s)$. The agent uses 16 options in total. Intuitively, an agent which has fewer option choices at a decision point should learn faster, since it has fewer alternatives to explore, but in the long run, this limits the space of usable policies for the agent. Fig. 1 confirms this trade-off between speed of learning and ultimate quality. Note that this trade-off holds the same way in planning as well (as discussed extensively in classical planning works).
Algorithm 1: IOC with tabular intra-option Q-learning

Initialize policy over options $\pi_O$
Initialize $I_{\omega,z}$ parameterized by $z$ such that all options are available everywhere to some extent
Initialize $\pi_{I_{\omega,z}}(\omega|s)$ as in Eq.(1)
Set $s \leftarrow s_0$ and $\omega$ at $s$ according to $\pi_{I_{\omega,z}}$
Repeat
  Choose $a$ according to $\pi_{\omega,\theta}(a|s)$
  Take action $a$ in $s$, observe $s', r$
  Sample termination from $\beta_{\omega,\nu}(s')$
  If $\omega$ terminates in $s'$ then
    Sample $\omega'$ according to $\pi_{I_{\omega,z}}(\cdot|s')$
  Else
    $\omega' = \omega$
  End if  
  
1. Evaluation step: 
$\delta \leftarrow r - Q_U(s, \omega, a)$
$\delta \leftarrow r + \gamma(1 - \beta_{\omega,\nu}(s'))Q_{\Omega}(s', \omega) + \gamma\beta_{\omega,\nu}(s')\max_{\omega'} Q_{\Omega}(s', \omega')$
$Q_U(s, \omega, a) \leftarrow Q_U(s, \omega, a) + \alpha\delta$

2. Improvement step 
$\theta \leftarrow \theta + \alpha\theta \frac{\partial \log \pi_{\omega,\theta}(a|s)}{\partial \theta} Q_U(s, \omega, a)$
$\nu \leftarrow \nu - \alpha\nu \frac{\partial \beta_{\omega,\nu}(s')}{{\nu}}(Q_{\Omega}(s', \omega) - V_{\Omega}(s'))$ where $V_{\Omega}(s') = \sum_{\omega'} \pi_{I_{\omega,z}}(\omega'|s')Q_{\Omega}(s', \omega')$
$z \leftarrow z + \alpha z \frac{\partial \beta_{\omega,\nu}(s')}{{\nu}}(Q_{\Omega}(s', \omega') - V_{\Omega}(s'))$
$s \leftarrow s'$
 Until $s'$ is a terminal state

5 Experimental Results

We now study the empirical behavior of IOC in order to answer the following questions: (1) are options with interest functions useful in a single task; (2) do interest functions facilitate learning reusable options and, (3) do interest functions provide better interpretability of the skills of an agent.

5.1 Learning in a single task

To analyze the utility of interest functions when learning in a single task, consider a given, fixed policy over options, either specified by a just-in-time planner or via human input. This setup allows us to understand the impact of interest functions alone in the learning process.

Four rooms (FR) We first consider the classic FR domain [36] (Fig. 3(a)). The agent starts at a uniformly random state and there is a goal state in the bottom right hallway ($\gamma = 0.9$). The environment is stochastic. The reward is +50 at the goal and 0 otherwise. We used 4 options,
whose intra-option policies were parameterized with Boltzmann distributions, and termination and interest functions represented as linear-sigmoid functions. Options were learned using either IOC or OC with tabular intra-option Q-learning, as described in Algorithm 1. Learning proceeds for 500 episodes, with a maximum of 2000 time steps allowed per episode. Additional details are provided in Appendix A.3.1.

**Results:** Fig. 2(a) shows the steps to goal for both OC and IOC, averaged over 70 independent runs. The IOC agent performs better than OC agent by roughly 100 steps. One potential reason for the improvement in IOC is that options become specialized to different regions of the state-space, as can be seen in Fig. 3. We also observe that the termination functions (which were initialized to 0) naturally become coherent with the interest functions learned, and are mostly room specific for each option (see Fig. A1). On the other hand, options learned by OC do not show such specialization and terminate everywhere (see Fig. A1). These results demonstrate that the IOC agent is not only able to correct for the given higher level policy, but also, leads to more understandable options as a side effect.

**TMaze** Next, we illustrate the learning and use of interest functions in the non-linear function approximation setting, using simple continuous control tasks implemented in Mujoco [38]. A point mass agent (blue) is located at the bottom end of a T-shaped maze (0, −0.1) and must navigate to within 0.1 of the goal position (0.3, 0.3) at the right end of the maze (green location) (Fig. 4(e)). The state space consists of the $x,y$ coordinates of the agent and the action space consists of force applied in the $x,y$ directions. We use a uniform fixed policy over options for both IOC and OC. We reuse the Proximal Policy Option-Critic (PPOC) algorithm [14] and add a 2-layer network with sigmoid outputs to compute the interest functions. However, we do correct the implementation of the gradient of the policy over options which has been overlooked in that work. The remaining update rules are consistent with Algorithm 1. Complete details about the implementation and hyper-parameter search are provided in Appendix A.3.2.

**Results:** We report the average performance over 10 independent runs. The IOC agent is able to converge in almost half the time steps needed by the OC agent. Potentially, interest functions in the IOC agent provide an attention mechanism and thus facilitates learning options which are more diverse (see Fig. 5 for evidence). A deeper analysis of interest functions learned in this domain is deferred to Sec. 5.3.

**MiniWorld** We also explore learning in more complex 3D first-person visual environment from the MiniWorld framework [4]. We use the Oneroom task where the agent has to navigate to a randomly placed red block in a closed room (Fig. 4(f)). This requires the agent to turn around and scan the room to find the red box, then navigate to it.

The observation space is a 3-channel RGB image and the action space consists of 8 discrete actions. At the start of each episode, the red box is placed randomly. The episode terminates if the agent reaches the box or max of 180 time steps is reached. We used the DQN architecture of [27]. See Appendix A.3.3 for details about implementation and hyper-parameters.

**Results:** The IOC agent is able to converge much faster (100 iterations) than the OC agent with a given uniform policy over option (Fig. 2(c)). The performance is averaged across 10 runs.

Based on these experiments, IOC provides improvement in performance consistently across a range of tasks, from the simple four-rooms domain to complex visual navigation tasks such as MiniWorld, indicating the utility of learning interest functions.

---

1 In this work we name this algorithm simply OC for option-critic.
Figure 4: **Transfer in continual learning setting:** Continuous Control in TMaze: (d) The point mass agent (blue) has two goals (green) both resulting in a reward of +1. After 150 iterations, the goal that is most visited is removed (e). IOC converges fastest in the first few iterations (a). After the task change, IOC suffers the least in terms of immediate loss in performance and gets the best final score. Visual navigation in Miniworld: requires the agent to go to a randomly placed red box in a closed room. After 150 episodes the agent has to navigate to an unseen blue box (g). IOC quickly adapts to the change (b) indicating that harnessing options learned from the old task speeds up learning in the new task. Locomotion in HalfCheetah: The cheetah is rewarded for moving forward during the first 150 iterations, after which it is rewarded for going backwards as fast as possible.

### 5.2 Option reusability

One of the primary reasons for an agent to autonomously learn options is the ability to generalize its knowledge quickly in new tasks. We now evaluate our approach in settings where adaptation to changes in the task is vital.

**TMaze** The point mass agent starts at the bottom of the maze, with two goal locations (Fig. 4(d)), both giving a reward of +1. After 150 episodes, the goal that has been visited the most is removed and the agent has to adapt its policy to the remaining goal available (Fig. 4(e)). Both OC and IOC learn 2 options. We use a softmax policy over options for both IOC and OC, which is also learned at the same time.

**Results:** In the initial phase, the difference in performance between IOC and the other two agents (OC and PPO) is striking (Fig. 4(a)): IOC converges twice as fast. Moreover, when the most visited goal is removed and adaptation to the task change is required, the IOC agent is able to explore faster and its performance declines less. This suggests that the structure learned by IOC provides more generality. At the end of task 2, IOC recovers its original performance, whereas PPO fails to recover during the allotted learning trials.

**MiniWorld** Initially, the agent is tasked to search and navigate to a randomly placed red box in one closed room (Fig. 4(f)). After 150 episodes, the agent has to adapt its skills to navigate to a randomly located blue box (Fig. 4(g)) which it has never seen before. Here, the policy over options as well as all the option components are being learned at the same time.

**Results:** The IOC agent outperforms both OC and PPO agents when required to adapt to the new task (Fig. 4(b)). This result indicates that the options learned with interest functions are more easily transferable. The IOC agent is able to adapt faster to unseen scenarios.

**HalfCheetah** We also study adaptation in learning a complex locomotion task for a planar cheetah. The initial configuration of this environment follows the standard HalfCheetah-v1 from OpenAI’s Gym: the agent is rewarded for moving forward as fast as possible. After 150 iterations, we modify the reward function so that the agent is now encouraged to move backward as fast as possible [9].

**Results:** PPO outperforms both OC and IOC in the initial task. However, as soon as the task changes, IOC reacts in the most efficient way and converges to the highest score after 500 iterations (Fig. 4(c)).
As seen consistently in all the environments, IOC generalizes much better over tasks, whereas PPO seems to overfit to the first task and generalizes poorly when the task changes.

In all our experiments, we notice that interest functions result in option specialization, which leads to both reusability and adaptability (i.e. an option may get slightly tweaked), especially in the complex tasks.

5.3 Option interpretability

To gain a better understanding of the agent’s behavior, we visualize different aspects of the learning process in several tasks.

**TMaze** We visualize the interest functions learned in TMaze (Fig. 5). Initially, the interest functions are randomized. At the end of the first task, the interest function for option 0 specializes in the lower diagonal of the state-space (Fig. 5(b)), whereas option 1’s interest function is completely different (Fig. 5(c)). When the task changes, the options readjust their interest. Eventually, the interest functions for the two options automatically specialize in different regions of the state space (last column of Fig. 5(c) & 5(b)). Fig. 5(a) illustrates the agent trajectories at different time instances, where the yellow and red dots indicate the two different options during the trajectory. A visualization of the emergence of interest functions during the learning process is also available on the project page. In contrast, the options learned by the OC agent are employed everywhere and have not specialized as much (see Appendix Fig. A2).

![Figure 5: Qualitative analysis of IOC in TMaze](https://sites.google.com/view/optionsofinterest)

**HalfCheetah** We analyze the skills learned in HalfCheetah. During the task of moving forward as fast as possible, the IOC agent employs option 0 to move forward by dragging its limbs, and option 1 to take much larger hopped steps (Fig. 6). Fig. 6 demonstrates the emergence of these very distinct skills and the agent’s switching between them across time. Additionally, we analyzed each option at the end of task 2 in which the agent was rewarded for moving backward. Option 0 now specializes in moving forward while option 1 focuses on moving backward. This is nice, as the agent preserves some ability to now solve both tasks. OC doesn’t learn options which are as distinct, and both options end up going backward and overfitting to the new task (see 2).

**MiniWorld** We visualize the skills acquired by inspecting the agent’s behavior at the end of first task. The IOC agent has learned two distinct options: option 0 scans the surroundings, whereas option 1

---

7

---

https://sites.google.com/view/optionsofinterest
is used to directly navigate towards the block upon successfully locating it (Fig. 7). During task 2, option 1 is being harnessed primarily to move forward, whereas option 0 is employed when jittery motion is involved, such as turning and scanning.

6 Related Work

Temporal abstraction in RL has a rich history [28, 37, 5, 6, 25, 26, 33]. Options in particular have been shown to speed up convergence both empirically [30] and theoretically [22]. Constructing such temporal abstractions automatically from data has also been tackled extensively, and with some success [16, 23, 18, 21, 3, 20]. While some of the approaches require prior knowledge, have a fixed time horizon for partial policies [39], or use intrinsic rewards [18], Bacon et al. [3] provides an end-to-end differentiable approach without needing any sub-goals or intrinsic motivation. We generalize their policy gradient approach to learn interest functions. While we use rewards in our gradient-based algorithm, our qualitative analysis also indicates some clustering of states in which a given option starts, as in [19, 1, 24]. Our approach is closely related in motivation to Mankowitz et al. [21]. However, our method does not make assumptions about a particular structure for the initiation and termination functions (except smoothness).

Initiation sets were an integral part of Sutton et al. [36] and provide a way to control the complexity of the process of exploration and planning with options. This aspect of options has been ignored since, including in recent works [3, 10–12] because there was no elegant way to learn initiation sets. We address this open problem by generalizing initiation sets to differentiable interest functions. Since an interest function is a component of an option, it can be transferred once learned.

7 Discussion and Future Directions

We introduced the notion of interest functions for options, which generalize initiation sets, with the purpose of controlling search complexity. We presented a policy gradient-based method to learn options with interest functions, using general function approximation. Because the learned options are specialized, they are able to both learn faster in a single task and adapt to changes much more efficiently than options which initiate everywhere. Our qualitative results suggest that the interest function could be interpreted as an attention mechanism (see Appendix Fig. A4). To some extent, the interest functions learnt are able to override termination degeneracy as well (only one option being active all the time, or options switching often) although our approach was not meant to tackle that problem directly. Exploring further the interaction of initiation and termination functions, and imposing more coordination between the two, is an interesting topic for future work.

In our current experiments, the agent optimizes a single external reward function. However, the same algorithm could be used with intrinsic rewards as well.
We did not explore in this paper the impact of interest functions in the context of planning. However, given the intuitions from classical planning, learning models for options with interest functions could lead to better and faster planning, which should be explored in the future.

Finally, other ways of incorporating interest functions into the policy over options would be worth considering, in order to consider only choices over few options at a time.

Acknowledgments

The authors would like to thank NSERC & CIFAR for funding this research; Emmanuel Bengio, Kushal Arora for useful discussions throughout this project; Michael Littman, Zafarali Ahmed, Nishant Anand, and the anonymous reviewers for providing critical and constructive feedback.

References


Options of Interest: Temporal Abstraction with Interest Functions

A Appendix

A.1 Reproducibility Checklist

We follow the reproducibility checklist by [29] and point to relevant sections explaining them here.

For all algorithms presented, check if you include:

- **A clear description of the algorithm.** See main paper Algorithm 1. Also see included codebase provided as a zip file.
- **An analysis of the complexity (time, space, sample size) of the algorithm.** We do not perform a complexity analysis of the algorithm.
- **A link to a downloadable source code, including all dependencies.** See experimental section in Appendix and main paper, the code is included with supplemental as zip file also.

For any theoretical claim, check if you include:

- **A statement of the result.** See main paper and Proofs section in the appendix A.2.
- **A clear explanation of any assumptions.** See Proofs section in the appendix A.2.
- **A complete proof of the claim.** See Proofs section in the appendix A.2.

For all figures and tables that present empirical results, check if you include:

- **A complete description of the data collection process, including sample size.** We use custom variants of the tasks in standard benchmarks such as Mujoco [38] and MiniWorld [4]. We provide the code for the same in the supplementary material.
- **A link to a downloadable version of the dataset or simulation environment.** See [4] and [38] for standard versions of these benchmarks. For fourrooms domain and custom changes to the tasks in Mujoco and MiniWorld see the the code is included with supplementary material.
- **An explanation of any data that were excluded, description of any pre-processing step.** We did not exclude any data in the environments we used. We do not require any data pre-processing step for our experiments.
- **An explanation of how samples were allocated for training / validation / testing.** We do not use a split as we are examining the optimization performance. Therefore, we report the performance during the learning process as shown in the figures. Once the model has been trained, we load the stored weights to demonstrate the performance via the videos and images as well.
- **The range of hyper-parameters considered, method to select the best hyper-parameter configuration, and specification of all hyper-parameters used to generate results.** Besides maintaining consistency the default values of the baselines, we optimize both the baseline and our algorithm for other parameters. See Experimental Details for complete implementation and hyper-parameter details in Appendix Sec A.3.
- **The exact number of evaluation runs.** We used 10 independent seeds for TMaze experiments, 10 independent runs for MiniWorld first set of experiments, and 5 independent seeds for remaining experiments of HalfCheetah and MiniWorld. For tabular, we averaged performance across 70 independent runs. We did not run for more number of seeds due to time and computational constraints.
- **A description of how experiments were run.** See Experimental Results Sec 5 in the main paper and for additional details see Appendix Sec A.3.
- **A clear definition of the specific measure or statistics used to report results.** We average discounted returns across all seeds as reported in the performance curves.
• **Clearly defined error bars.** We report the standard error in all cases.

• **A description of results with central tendency (e.g. mean) & variation (e.g. stddev).** We report the standard error in all cases. All figures with the returns show the standard error across the independent random seeds.

• **A description of the computing infrastructure used.** For tabular experiments, we use 1 CPU. For function approximation, we distribute all runs across 1 CPU and 1 GPU per run. Our CNN code takes longer to run (10 hours/run) as compared to our TMaze, HalfCheetah code (2 hours/run).

### A.2 Proofs

#### A.2.1 Proof of Interest Function Gradient Theorem

**Theorem.** Given a set of Markov options with differentiable interest functions $I_{\omega,z}$, where $z$ is the parameter vector; the gradient of the expected discounted return with respect to $z$ at $(s, \omega)$ is:

$$
\sum_{s', \omega'} \hat{\mu}_\omega(s', \omega'|s, \omega) \beta_\omega(s') \frac{\partial I_{\omega,z}(\omega'|s)}{\partial z} Q_\omega(s', \omega')
$$

where $\hat{\mu}_\omega(s', \omega'|s, \omega)$ is the discounted weighting of the state-option pairs along trajectories starting from $(s, \omega)$ sampled from the distribution determined by $\pi_{I_{\omega,z}}$, $\beta_\omega$ is the termination function and $Q_\omega$ is the value function over options corresponding to $\pi_{I_{\omega,z}}$.

**Proof.** Let us start with the option-value function; $Q_{\Omega, \theta}(s, \omega)$

$$
Q_{\Omega}(s, \omega) = \sum_a \pi_{\omega, \theta}(a|s)Q_U(s, \omega, a)
$$

which depends on the interest function $I$ which is parameterized by $z$. Therefore taking the derivation of $Q_{\Omega}$ w.r.t $z$.

$$
\frac{\partial Q_{\Omega}(s, \omega)}{\partial z} = \frac{\partial}{\partial z} \left\{ \sum_a \pi_{\omega, \theta}(a|s)Q_U(s, \omega, a) \right\}
$$

Expanding $Q_U$; we get

$$
\frac{\partial Q_{\Omega}(s, \omega)}{\partial z} = \frac{\partial}{\partial z} \left\{ \sum_a \pi_{\omega, \theta}(a|s) \left( r(s, a) + \gamma \sum_{s'} P(s'|s, a)U(\omega, s') \right) \right\}
$$

$$
= \frac{\partial}{\partial z} \left\{ \sum_a \pi_{\omega, \theta}(a|s) \left( r(s, a) + \gamma \sum_{s'} P(s'|s, a) \left( (1 - \beta_{\omega, \nu}(s'))Q_\omega(s', \omega) + \beta_{\omega, \nu}(s')V_{\omega}(s') \right) \right) \right\}
$$

$$
= \sum_a \pi_{\omega, \theta}(a|s) \sum_{s'} \gamma P(s'|s, a) \left( (1 - \beta_{\omega, \nu}(s')) \frac{\partial Q_\omega(s', \omega)}{\partial z} + \beta_{\omega, \nu}(s') \frac{\partial V_\omega(s')}{\partial z} \right)
$$

Now the state-value function can be expressed in terms of option-value function as follows:

$$
V_{\Omega}(s) = \sum_{\omega} \pi_{I_{\omega,z}}(\omega|s)Q_{\Omega}(s, \omega)
$$

where

$$
\pi_{I_{\omega,z}}(\omega|s) = I_{\omega,z}(s)\pi_{\Omega}(\omega|s) / \sum_{\omega'} I_{\omega', z}(s)\pi_{\Omega}(\omega'|s)
$$

Note that $\pi_{\Omega}(\omega|s)$ is fixed and not parameterized here.

Then, taking gradient w.r.t $z$ yields:

$$
\frac{\partial V_{\Omega}(s')}{\partial z} = \sum_{\omega} \left( \frac{\partial \pi_{I_{\omega,z}}(\omega|s')}{\partial z} Q_{\Omega}(s', \omega) + \pi_{I_{\omega,z}}(\omega|s') \frac{\partial Q_{\Omega}(s', \omega)}{\partial z} \right)
$$
Substituting (5) in (3), we get:

\[
\frac{\partial Q_\Omega(s, \omega)}{\partial z} = \sum_a \pi_{\omega, \theta}(a|s) \sum_{s'} \gamma P(s'|s, a) \left( (1 - \beta_{\omega, \nu}(s')) \frac{\partial Q_\Omega(s', \omega)}{\partial z} + \beta_{\omega, \nu}(s') \right)
\]

Collecting coefficients together:

\[
\frac{\partial Q_\Omega(s, \omega)}{\partial z} = \sum_a \pi_{\omega, \theta}(a|s) \sum_{s'} \gamma P(s'|s, a) \sum_{\omega'} \beta_{\omega, \nu}(s') \frac{\partial \pi_{I_{\omega, z}}(\omega'|s')}{\partial z} Q_\Omega(s', \omega') + \sum_a \pi_{\omega, \theta}(a|s) \gamma P(s'|s, a) \sum_{s', \omega'} \left( (1 - \beta_{\omega, \nu}(s')) + \beta_{\omega, \nu}(s') \pi_{I_{\omega, z}}(\omega'|s') \right) \frac{\partial Q_\Omega(s', \omega')}{\partial z}
\]

Rearranging summations for term 2, we get:

\[
\frac{\partial Q_\Omega(s, \omega)}{\partial z} = \sum_a \pi_{\omega, \theta}(a|s) \sum_{s'} \gamma P(s'|s, a) \sum_{\omega'} \beta_{\omega, \nu}(s') \frac{\partial \pi_{I_{\omega, z}}(\omega'|s')}{\partial z} Q_\Omega(s', \omega') + \sum_{s'} \sum_{\omega'} \left( \sum_a \pi_{\omega, \theta}(a|s) \gamma P(s'|s, a) \left( (1 - \beta_{\omega, \nu}(s')) + \beta_{\omega, \nu}(s') \pi_{I_{\omega, z}}(\omega'|s') \right) \right) \frac{\partial Q_\Omega(s', \omega')}{\partial z}
\]

In the above equation, one-step discounted transition probability in the augmented space is given as

\[
P^{(1)}_\gamma(s', \omega'|s, \omega) = \sum_a \pi_{\omega, \theta}(a|s) \gamma P(s'|s, a) \left( (1 - \beta_{\omega, \nu}(s')) \mathbb{I}_{\omega = \omega'} + \beta_{\omega, \nu}(s') \pi_{I_{\omega, z}}(\omega'|s') \right)
\]

Thus we rewrite \(\frac{\partial Q_\Omega(s, \omega)}{\partial z}\) as:

\[
\frac{\partial Q_\Omega(s, \omega)}{\partial z} = \sum_a \pi_{\omega, \theta}(a|s) \sum_{s'} \gamma P(s'|s, a) \sum_{\omega'} \beta_{\omega, \nu}(s') \frac{\partial \pi_{I_{\omega, z}}(\omega'|s')}{\partial z} Q_\Omega(s', \omega') + \sum_{s'} \sum_{\omega'} P^{(1)}_\gamma(s', \omega'|s, \omega) \frac{\partial Q_\Omega(s', \omega')}{\partial z}
\]

Since we have recursion between current \(s, \omega\) with consecutive state-option pairs, using the \(k\)-steps augmented process as shown in Section A.2.4, we obtain the following:

\[
\frac{\partial Q_\Omega(s, \omega)}{\partial z} = \sum_{k=0}^\infty \sum_{s', \omega'} P^{(k)}_\gamma(s', \omega'|s, \omega) \sum_a \pi_{\omega, \theta}(a|s') \beta_{\omega, \nu}(s') \frac{\partial \pi_{I_{\omega, z}}(\omega'|s')}{\partial z} Q_\Omega(s', \omega')
\]

(6)

Therefore, the interest-function gradient is given as follows:

\[
= \sum_{s', \omega'} \bar{\mu}_\Omega(s', \omega'|s, \omega) \beta_{\omega, \nu}(s') \frac{\partial \pi_{I_{\omega, z}}(\omega'|s')}{\partial z} Q_{\Omega, \theta}(s', \omega')
\]

(7)

where \(\bar{\mu}_\Omega(s', \omega'|s, \omega)\) is the discounted weighting of the state-option pairs along trajectories starting from \((s, \omega)\) sampled from the distribution determined by \(\pi_{I_{\omega, z}}\), \(\bar{\mu}_\nu\) is the termination function and \(Q_{\Omega, \theta}\) is the value function over options corresponding to \(\pi_{I_{\omega, z}}\). Note that this differs from the discounted weighting of state-option pairs in the option-critic derivation. For the interest function gradient update, the derivation of \(\pi_{I_{\omega, z}}(\omega|s)\) with respect to the parameter \(z\) is shown in Section A.2.5 in this appendix. 

\[\square\]
A.2.2 Proof of Intra-Option Policy Gradient Theorem

**Theorem.** Given a set of Markov options with stochastic, differentiable intra-option policies \( \pi_{\omega, \theta} \), the gradient of the expected discounted return with respect to \( \theta \) and initial condition \((s_0, \omega_0)\) is:

\[
\sum_{s, \omega} \hat{\mu}_\Omega(s, \omega|s_0, \omega_0) \sum_a \frac{\partial \pi_{\omega, \theta}(a|s)}{\partial \theta} Q_U(s, \omega, a)
\]

where \( \hat{\mu}_\Omega(s, \omega|s_0, \omega_0) \) is the discounted weighting of the state-option pairs along trajectories starting from \((s_0, \omega_0)\) sampled from the new option sampling distribution determined by \( I_{\omega, z}(s) \).

**Proof.** Starting with the option-value function \( Q_\Omega(s, \omega) \), we take the gradient of this with respect to \( \theta \) as follows:

\[
\frac{\partial Q_\Omega(s, \omega)}{\partial \theta} = \sum_a \left( \frac{\partial \pi_{\omega, \theta}(a|s)}{\partial \theta} Q_U(s, \omega, a) + \pi_{\omega, \theta}(a|s) \frac{\partial Q_U(s, \omega, a)}{\partial \theta} \right)
\]

\[
= \sum_a \left( \frac{\partial \pi_{\omega, \theta}(a|s)}{\partial \theta} Q_U(s, \omega, a) + \pi_{\omega, \theta}(a|s) \sum_{s'} \gamma P(s'|s, a) \frac{\partial U(\omega, s')}{\partial \theta} \right)
\]

(8)

Now \( U(\omega, s') = (1 - \beta_{\omega, \nu}(s')) Q_\Omega(s', \omega) + \beta_{\omega, \nu}(s') V_\Omega(s') \). Substituting \( V_\Omega(s') \) and then taking the gradient of \( U \) with respect to \( \theta \), we get:

\[
\frac{\partial U(\omega, s')}{\partial \theta} = (1 - \beta_{\omega, \nu}(s')) \frac{\partial Q_\Omega(s', \omega)}{\partial \theta} + \beta_{\omega, \nu}(s') \frac{\partial Q_U(s', \omega, a)}{\partial \theta}
\]

\[
= \sum_{s'} \left[ (1 - \beta_{\omega, \nu}(s')) I_{\omega' = \omega} + \beta_{\omega, \nu}(s') \left( I_{\omega, z}(s') \pi_{\Omega}(\omega'|s') / \sum_{\omega'} I_{\omega, z}(s') \pi_{\Omega}(\omega|s') \right) \right] \frac{\partial Q_\Omega(s', \omega')}{\partial \theta}
\]

(9)

Substituting (9) in (8), we get:

\[
\frac{\partial Q_\Omega(s, \omega)}{\partial \theta} = \sum_a \frac{\partial \pi_{\omega, \theta}(a|s)}{\partial \theta} Q_U(s, \omega, a) + \sum_a \pi_{\omega, \theta}(a|s) \sum_{s'} \gamma P(s'|s, a) \times \sum_{s'} \left( (1 - \beta_{\omega, \nu}(s')) I_{\omega' = \omega} + \beta_{\omega, \nu}(s') \left( I_{\omega, z}(s') \pi_{\Omega}(\omega'|s') / \sum_{\omega'} I_{\omega, z}(s') \pi_{\Omega}(\omega|s') \right) \right) \frac{\partial Q_\Omega(s', \omega')}{\partial \theta}
\]

Substituting one-step discounted probability from the augmented process as shown in Section A.2.4, we get:

\[
= \sum_a \frac{\partial \pi_{\omega, \theta}(a|s)}{\partial \theta} Q_U(s, \omega, a) + \sum_{s', \omega'} \sum_{s'} P_{\gamma}^{(1)}(s', \omega'|s, \omega) \frac{\partial Q_\Omega(s', \omega')}{\partial \theta}
\]

By extension to k time steps, we get:

\[
= \sum_{k=0}^{\infty} \sum_{s', \omega'} P_{\gamma}^{(k)}(s', \omega'|s, \omega) \sum_a \frac{\partial \pi_{\omega, \theta}(a|s')}{\partial \theta} Q_U(s, \omega, a)
\]

\[
= \sum_{s, \omega} \hat{\mu}_\Omega(s, \omega|s_0, \omega_0) \sum_a \frac{\partial \pi_{\omega, \theta}(a|s)}{\partial \theta} Q_U(s, \omega, a)
\]

where \( \hat{\mu}_\Omega(s, \omega|s_0, \omega_0) = \sum_{t=0}^{\infty} \gamma^t P(s_t = s, \omega_t = \omega|s_0, \omega_0) \). Note that this differs from the discounted weighting of state-option pairs in the option-critic derivation. □
A.2.3 Proof of Termination-Gradient Theorem

**Theorem.** Given a set of Markov options with stochastic, differentiable termination functions \( \beta_{\omega,\nu} \), the gradient of the expected discounted return with respect to \( \nu \) and initial condition \((s_1, \omega_0)\) is:

\[
- \sum_{s', \omega} \hat{\mu}_\Omega(s', \omega|s_1, \omega_0) \sum_a \frac{\partial \beta_{\omega,\nu}(s')}{\partial \nu} A_\Omega(s', \omega)
\]

where \( \hat{\mu}_\Omega(s, \omega|s_0, \omega_0) \) is the discounted weighting of the state-option pairs along trajectories starting from \((s_0, \omega_0)\) sampled from the new option sampling distribution determined by \( I_{\omega,z}(s) \).

**Proof.** The proof is fairly simple and follows through similar to [3]. The only and the key difference in the result is the new discounted weighting of state-option pairs as shown in Section A.2.4. \( \hat{\mu}_\Omega(s, \omega|s_0, \omega_0) \) now includes interest functions influencing how policy over options are chosen in any given state as opposed to all options being present everywhere.

This time, we work with the expected sum of discounted rewards starting from \((s_1, \omega_0)\):

\[
U(\omega_0, s_1) = E \left[ \sum_{t=1}^{\infty} \gamma^{t-1} r_t | s_1, \omega_0 \right]
\]

\( U \) can be expanded as follows:

\[
U(\omega, s') = (1 - \beta_{\omega,\nu}(s')) Q_{\Omega}(s', \omega) + \beta_{\omega,\nu}(s') V_{\Omega}(s')
\]

With interest functions, this is further expanded:

\[
U(\omega, s') = (1 - \beta_{\omega,\nu}(s')) Q_{\Omega}(s', \omega) + \beta_{\omega,\nu}(s') \sum_{\omega'} \left\{ I_{\omega',z}(s') \pi_{\Omega}(\omega|s') / \sum_{\omega} I_{\omega',z}(s') \pi_{\Omega}(\omega'|s') Q_{\Omega}(s', \omega') \right\}
\]

Expanding \( U \) now;

\[
= (1 - \beta_{\omega,\nu}(s')) \sum_{a} \pi_{\omega,\theta}(a|s) \left( r(s', a) + \sum_{s''} \gamma P(s''|s', a) U(\omega, s'') \right) + \beta_{\omega,\nu}(s') \sum_{\omega'} \left\{ I_{\omega',z}(s') \pi_{\Omega}(\omega'|s') / \sum_{\omega} I_{\omega',z}(s') \pi_{\Omega}(\omega'|s') \right\} \times \sum_{a} \pi_{\omega',\theta}(a|s') \left( r(s', a) + \sum_{s''} \gamma P(s''|s', a) U(\omega', s'') \right)
\]

Gradient of \( U \) then becomes:

\[
\frac{\partial U(\omega, s')}{\partial \nu} = \frac{\partial \beta_{\omega,\nu}(s')}{\partial \nu} (V_{\Omega}(s') - Q_{\Omega}(s', \omega)) + (1 - \beta_{\omega,\nu}(s')) \sum_{a} \pi_{\omega,\theta}(a|s') \sum_{s''} \gamma P(s''|s', a) U(\omega, s'')
\]

Substituting the advantage function \( A_{\Omega}(s', \omega) = Q_{\Omega}(s', \omega) - V_{\Omega}(s') \) and using the augmented process, we get:

\[
\frac{\partial U(\omega, s')}{\partial \nu} = - \frac{\partial \beta_{\omega,\nu}(s')}{\partial \nu} A_{\Omega}(s', \omega) + \sum_{\omega'} \sum_{s''} P_{\nu}(s'', \omega'|s', \omega) \frac{\partial U(\omega', s'')}{\partial \nu}
\]

Using the recursive form and extending to \( k \)-time steps, we get:

\[
= - \sum_{\omega', s''} \sum_{k=0}^{\infty} P_{\nu}^{(k)}(s'', \omega'|s', \omega) \frac{\partial \beta_{\omega,\nu}(s'')}{\partial \nu} A_{\Omega}(s'', \omega')
\]

This gives us the following result:

\[
\frac{\partial U(\omega_0, s_1)}{\partial \nu} = - \sum_{s', \omega} \hat{\mu}_\Omega(s', \omega|s_1, \omega_0) \frac{\partial \beta_{\omega,\nu}(s')}{\partial \nu} A_{\Omega}(s', \omega)
\]

where \( \hat{\mu}_\Omega(s, \omega|s_0, \omega_0) = \sum_{t=0}^{\infty} \gamma^t P(s_t = s, \omega_t = \omega|s_0, \omega_0) \). Note that this differs from the discounted weighting of state-option pairs in the option-critic derivation.
A.2.4 Augmented Process with Interest Functions

Following the augmented process shown in [3], we present how the new formulation of option-critic with interest functions impacts this process. Let us consider that an option \( \omega_t \) has been initiated or is executing at time \( t \), then the discounted probability of transitioning to \( (s_{t+1}, \omega_{t+1}) \) is now given as:

\[
P^{(1)}_\gamma(s_{t+1}, \omega_{t+1}|s_t, \omega_t) = \sum_a \pi_{\omega_t, \theta}(a|s_t) \gamma P(s_{t+1}|s_t, a) \left( 1 - \beta_{\omega_t, \nu}(s_t) I_{\omega_t=\omega_t+1} + \beta_{\omega_t, \nu}(s_t) \left( I_{\omega_t+1, z_t}(s_{t+1}) \pi_\Omega(\omega_{t+1}|s_{t+1}) \sum_\omega I_{\omega, z}(s_{t+1}) \pi_\Omega(\omega|s_{t+1}) \right) \right)
\]

When transitioning from \( (s_t, \omega_t-1) \rightarrow (s_{t+1}, \omega_t) \), the discounted probability is:

\[
P^{(1)}_\gamma(s_{t+1}, \omega_t|s_t, \omega_t-1) = (1 - \beta_{\omega_t-1, \nu}(s_t)) I_{\omega_t=\omega_t-1} + \beta_{\omega_t-1, \nu}(s_t) \times \left( I_{\omega_t, z_t}(s_{t+1}) \pi_\Omega(\omega_t|s_{t+1}) \sum_\omega I_{\omega, z}(s_{t+1}) \pi_\Omega(\omega|s_{t+1}) \right) \sum_a \pi_{\omega_t, \theta}(a|s_t) \gamma P(s_{t+1}|s_t, a)
\]

More generally, the \( k \)-steps discounted probability can be expressed recursively as follows:

\[
P^{(k)}_\gamma(s_{t+k}, \omega_{t+k}|s_t, \omega_t) = \sum_{s_t+1} \sum_{s_{t+1}} \left( P^{(1)}_\gamma(s_{t+1}, \omega_{t+1}|s_t, \omega_t) P^{(k-1)}_\gamma(s_{t+k}, \omega_{t+k+1}|s_{t+1}, \omega_{t+1}) \right)
\]

\[
P^{(k)}_\gamma(s_{t+k}, \omega_{t+k-1}|s_t, \omega_t) = \sum_{s_t+1} \sum_{s_{t+1}} \left( P^{(1)}_\gamma(s_{t+1}, \omega_{t}|s_t, \omega_{t-1}) P^{(k-1)}_\gamma(s_{t+k}, \omega_{t+k-1}|s_{t+1}, \omega_{t}) \right)
\]

The augmented process with the introduction of Interest Functions turns out to be same as in [3] with the only and main difference in how policy over options are selected in any given state is now determined as following: \( \pi_{I_{\omega, z}}(\omega|s) = I_{\omega, z}(s) \pi_\Omega(\omega|s) \sum_\omega I_{\omega, z}(s) \pi_\Omega(\omega|s) \).

A.2.5 Derivation of \( \pi_{I_{\omega, z}}(\omega|s) \)

In this section, we show the derivation of the interest-functions induced probability distribution for sampling options in any given state:

\[
\pi_{I_{\omega, z}}(\omega|s) = I_{\omega, z}(s) \pi_\Omega(\omega|s) / \sum_\omega I_{\omega, z}(s) \pi_\Omega(\omega|s)
\]

Using the log-trick and taking gradient with respect to \( z \), we could rewrite it as:

\[
\nabla_z \pi_{I_{\omega, z}}(\omega|s) = \pi_{I_{\omega, z}}(\omega|s) \nabla_z \log \pi_{I_{\omega, z}}(\omega|s)
\]

Substituting the value of \( \pi_{I_{\omega, z}}(\omega|s) \)

\[
\pi_{I_{\omega, z}}(\omega|s) \nabla_z \log \left( I_{\omega, z}(s) \pi_\Omega(\omega|s) / \sum_\omega I_{\omega, z}(s) \pi_\Omega(\omega|s) \right)
\]

Using log-trick and taking gradient with respect to \( z \), we could rewrite it as:

\[
\nabla_z \pi_{I_{\omega, z}}(\omega|s) = \pi_{I_{\omega, z}}(\omega|s) \nabla_z \left( \log I_{\omega, z}(s) \pi_\Omega(\omega|s) - \log \pi_{I_{\omega, z}}(\omega|s) \right)
\]

Using log-trick and taking gradient with respect to \( z \), we could rewrite it as:

\[
\pi_{I_{\omega, z}}(\omega|s) \left( \nabla_z \log I_{\omega, z}(s) + \nabla_z \log \pi_\Omega(\omega|s) - \nabla_z \log \left( \sum_\omega I_{\omega, z}(s) \pi_\Omega(\omega|s) \right) \right)
\]

\[
= \pi_{I_{\omega, z}}(\omega|s) \nabla_z \log I_{\omega, z}(s) + \pi_{I_{\omega, z}}(\omega|s) \nabla_z \log \pi_\Omega(\omega|s) - \pi_{I_{\omega, z}}(\omega|s) \nabla_z \log \left( \sum_\omega I_{\omega, z}(s) \pi_\Omega(\omega|s) \right)
\]

The second term is equal to 0, so we obtain the following result:

\[
\pi_{I_{\omega, z}}(\omega|s) = \frac{1}{I_{\omega, z}(s)} \nabla_z I_{\omega, z}(s) - \pi_{I_{\omega, z}}(\omega|s) \frac{1}{\sum_\omega I_{\omega, z}(s) \pi_\Omega(\omega|s)} \sum_\omega \nabla_z I_{\omega, z}(s) \pi_\Omega(\omega|s)
\]
A.3 Experimental Details

A.3.1 Four Rooms Domain

Implementation details: The discount factor is 0.99, and the reward is +50 at the goal and 0 otherwise. We used 4 options, whose intra-option policies were parameterized with Boltzmann distributions, and termination and interest functions represented as linear-sigmoid functions. Options were learned using either Interest-Option-Critic (IOC) or Option-Critic (OC) with tabular intra-option Q-learning, as described in Algorithm 1.

Hyper-parameter details: Based on the values reported in [2], we used a baseline for the gradient estimator and a temperature of 0.01 for the intra-option policies, a learning rate of 0.5 for the critic and 0.25 for the termination and intra-option updates for both OC and IOC. For the IOC agent, a learning rate of 0.15 was used for the interest function updates. This was picked to be lower than the learning rate of $\beta$ chosen based on a small search. Learning proceeds for a total of 500 episodes, with a maximum of 2000 time steps allowed per episode. Interest functions weights were initialized with a room specific structure prior. All other weights are initialized to zeros. We ran 70 independent runs and average returns and steps across these runs. The code is provided with the supplementary material in a zip file.

Here we show a qualitative analysis of the options learned by the IOC and the OC agent (Fig. A1).

![Visualization of Interest Functions (a-d), IOC (e-h) and OC (i-l) Termination Conditions at the end of 500 episodes in task 1 with 4 options. Brighter colors represent higher values. Option learned with interest functions emerge with specific interest in different regions of the state space as shown in Figures (a) to (d). Each row represents Option 1 to 4 going from left to right. IOC Termination conditions for each option emerges complimenting the interest of that option as opposed to the termination conditions (i-l) for each option in OC which assumes that all options are available everywhere and therefore result in options terminating everywhere.](image-url)
A.3.2 TMaze

Implementation details: For function approximation, we built on top of the Proximal Policy Option-Critic (PPOC) algorithm [14] incorporating learning interest functions. The update rules are consistent with Algorithm 1. To represent the interest functions, we add another network to the PPOC algorithm: a 2-layer feed forward network with tanh activation. The input of the network is the state while the output is a sigmoid with size equal to the number of options, representing the interest for each option at the state. For the termination function, intra-option policies and state-option value functions, we keep the architecture consistent to Klissarov et al. [14]. We train for 150 iterations and average performance over 10 independent runs of the algorithm. The code is also provided with the supplementary material in a zip file.

Hyper-parameter details - TMaze Domain: For each algorithm, we report results for the best hyperparameters configuration, after performing a sweep on the intra-option policies, value function, termination function and interest function learning rates (Tab. 1). For the transfer experiments: In addition to the learning rates of interest function, intra-option policies, a sweep across learning rate of policy over option was also performed and the optimum parameters reported for both OC and IOC agents. All other parameters are kept consistent with the baseline.

<table>
<thead>
<tr>
<th>πω lr</th>
<th>1e−4</th>
<th>3e−4</th>
<th>5e−4</th>
<th>7e−4</th>
<th>9e−4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iω,z lr</td>
<td>1e−4</td>
<td>3e−4</td>
<td>5e−4</td>
<td>7e−4</td>
<td>9e−4</td>
</tr>
<tr>
<td>πΩ lr</td>
<td>1e−4</td>
<td>3e−4</td>
<td>5e−4</td>
<td>7e−4</td>
<td>9e−4</td>
</tr>
<tr>
<td>Seed</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>..</td>
<td>9</td>
</tr>
</tbody>
</table>

Figure A2: Analysis of Options learned by the OC agent Row 1: depicts sampled trajectories with Option 0 indicated by red dots and Option 0 by yellow dots. Row 2 & 3: show the policy over options which is initialized at random initially. Over time, the options learned by the OC agent have not achieved any specialization.
Analysis of options learned by the OC agent: We also investigated the options learned by the OC agent. Fig. A2 illustrates the timeline from the beginning of task 1 to the end of task 2. Since options in OC do not have interest functions, we visualize the learned policy over options to analyze the nature of options learned. Over time, the options learned by the OC agent have not achieved any specialization. This is a direct consequence of the assumption in OC that all options are available everywhere. Upon inspecting the sampled trajectories at the end of task 1 & 2, it is observed that the options cannot be interpreted as skills which have specialization in different regions of the state space. On the contrary, the interest of each options in the IOC agent lends meaning and focus to each option as depicted in the Fig. 5.

A.3.3 MiniWorld

Implementation details: We use the Oneroom task where the agent has to navigate to a randomly placed red block in a closed room (Fig. 4(f)). This requires for the agent to turn around and scan the room to find the red box. The observation space is a 3-channel RGB image of 60 × 80 dimension. The action space consists of discrete 8 actions. At the start of each episode the red box is placed randomly in the closed room. The episode terminates if the agent reaches the red box or a max time steps of 180 is reached.

In our experiments, a deep convolutional neural network is employed as function approximator which takes in the state image as input and outputs the hidden layer. The interest and termination function of each option for states is parameterized by sigmoid functions, the output is linear representing the value functions and intra-option policies with softmax policy over option for both IOC and OC which is also being learnt alongside options. The CNN architecture is kept consistent with DQN [27].

Hyper-parameter details - Miniworld Domain: The baseline algorithm is PPOC [14] adapted for discrete actions. Performance is reported over 10 independent runs (for the first experiment with a uniform fixed policy over option) after a complete sweep over the intra-option policies, value function, termination function and interest function learning rates (Tab. 2). For the transfer experiments: in addition to the learning rates of interest function, intra-option policies, a sweep across learning rate of policy over option was also performed and the optimum parameters reported for both OC and IOC agents. Performance is reported over 5 independent runs in this case. Complete range of hyper-parameters swept are mentioned in Tab. 2. All other parameters are kept consistent with the baseline.

<table>
<thead>
<tr>
<th>Hyper-parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\pi_{\omega}$ lr</td>
<td>$3e-4$ $7e-4$ $1e-4$ $5e-4$ $2e-4$</td>
</tr>
<tr>
<td>$I_{\omega,z}$ lr</td>
<td>$1e-04$ $3e-03$ $8e-04$ $8e-05$ $5e-04$ $3e-04$ $9e-05$</td>
</tr>
<tr>
<td>$\pi_{\Omega}$ lr</td>
<td>$3e-4$ $7e-4$ $1e-4$ $5e-4$ $2e-4$</td>
</tr>
<tr>
<td>Seed</td>
<td>0 1 2 3 4 5</td>
</tr>
</tbody>
</table>

A.3.4 HalfCheetah

Implementation details: The experiments for HalfCheetah use the exact same implementation details as the TMaze domain A.3.2. The environment has been customized to reflect our task specifications. The code is provided in the supplementary zip file.

Hyper-parameter details: Tab 3 enlists the complete hyper-parameter details.

Analysis of options learned by the OC agent: Analogous to the timeline of options learned by the IOC agent, we visualize the timeline of OC options over an episode (Fig. A3). We observe that the options are noisy and often switch to the other choices available. Options learned by the OC agent cannot be interpreted as distinct skills.

Figure A3: Timeline of options used by OC agent in HalfCheetah where each option is represented by a distinct color (black & white).
Table 3: Hyper-parameter details - HalfCheetah domain

<table>
<thead>
<tr>
<th>$\pi_{\omega, lr}$</th>
<th>$3 \times 10^{-4}$</th>
<th>$7 \times 10^{-4}$</th>
<th>$1 \times 10^{-4}$</th>
<th>$5 \times 10^{-4}$</th>
<th>$9 \times 10^{-4}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_{\omega, lr}$</td>
<td>$1 \times 10^{-4}$</td>
<td>$3 \times 10^{-4}$</td>
<td>$5 \times 10^{-4}$</td>
<td>$7 \times 10^{-4}$</td>
<td>$9 \times 10^{-4}$</td>
</tr>
<tr>
<td>$\pi_{\Omega, lr}$</td>
<td>$3 \times 10^{-3}$</td>
<td>$7 \times 10^{-3}$</td>
<td>$1 \times 10^{-3}$</td>
<td>$5 \times 10^{-3}$</td>
<td>$9 \times 10^{-3}$</td>
</tr>
<tr>
<td>Seed</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

A.4 Interest as an Attention Mechanism

To further analyze the meaning of interest functions learned by the IOC agent; we overlaid the interest of the active option on the timeline of an episode in HalfCheetah (Fig. A4). Our findings of how interest of an option activates across trajectories indicates that interest function is not just limited to being a measure of where an option initiates. It could be interpreted as an attention mechanism for where an option should attend to. In doing so, it lends the option an indication of where to stop attending as well (analogous to termination). To some extent, the interest functions learnt are able to override the termination degeneracies (only one option being active all the time, or options switching often) even though our approach does not tackle that problem directly.

Figure A4: **Interest Function** overlaid on the timeline of an episode of HalfCheetah. Option 0 and 1 are depicted by black and white color respectively. The blue line shows the interest of the active option, which peaks everytime an option is active.