Learning hyperparameter η for Proximal Policy Option-Critic (PPOC) Algorithm Applied To Continuous Action Tasks.

Arsh Padda, Raj Sadaye, Rushikesh Sargar, Shubham Gondane

***Abstract*— We build on the results of the Proximal Policy Options Critic algorithm for continuous control tasks and aim to learn the characteristics of the deliberation cost (η) that controls selection of new options in the learning process of an agent. We use the MuJoCo environment to model continuous action tasks. Results on these tasks yield interesting results which imply that the choice of the deliberation cost value depends highly on the reward model of the environment.**

I. INTRODUCTION

PPOC combines the proximal policy optimization algorithm while also utilizing option critic architecture to learn intra-option policies as well as termination conditions. PPO alternates between sampling data through interaction with the environment, and optimizing a surrogate objective function using stochastic gradient ascent. The surrogate objective function contains clipped probability ratios, preventing an excessive shift in the probability distribution between updates. The option-critic architecture capable of learning both the internal policies and the termination conditions of options, in tandem with the policy over options, and without the need to provide any additional rewards or sub goals. A deliberation cost (η) is introduced, interpreted as a margin of how much better an option should be than the current option in order to replace it.

This paper reviews the various papers related to the algorithm and discusses the implementation of the algorithm and states the observation and results at the end.

II. RELATED WORK

A. *Trust Region Policy Optimization [2]*

Trust Region Policy Optimization (TRPO) is an iterative procedure for optimizing policies with the guarantee of monotonic improvement.

Over exploration in policy optimization methods can lead to state distribution becoming heavily altered by the change in policy. This can cause the agent to forget optimal routes discovered in the past during a continuous task. Due to unreliable rewards, the policy may undergo too many changes. To handle this we need to carefully craft policy updates and consider maximum expected discounted reward

We can define the expected return of another policy in terms of advantage over accumulated over timesteps.

New Expected Return = Old Expected Return + Expected Cumulative Sum of Discounted Advantage

This method uses an approximation for decreasing complexity by using the old state visitation distribution even for newer policies and focus only on improving the policy. This approximation will work for small steps taken towards policy improvement.

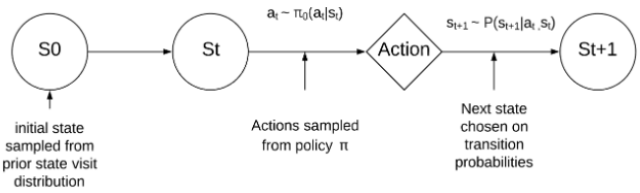

Fig. 1: Description of next state and action are sampled from initial state and prior state visitation distribution

B. *Actor Critic Algorithm [3]*

Value based methods maps each action state pair to a value (Think Value Function). Policy based methods optimizes the policy without using a value function (Think RL with Gradient Descent). Actor Critic: a Critic that measures how good the action taken is (value-based) and an Actor that controls how our agent behaves (policy-based) Actor Critic instead of waiting for an episode to end in order to get the rewards, we make improvement at each step.

Convergence of actor-critic algorithms: Since our actor-critic algorithms are gradient-based, one cannot expect to prove convergence to a globally optimal policy. The best that one could hope for is the convergence to zero; in practical terms, this will usually translate to convergence to a local minimum.

C. *Benchmarking Deep Reinforcement Learning for Continuous Control [4]*

Algorithms based on Q-learning quickly become infeasible when naive discretization of the action space is performed, due to the curse of dimensionality. The tasks in the presented benchmark can be divided into four categories: basic tasks, locomotion tasks, partially observable tasks, and hierarchical tasks

Trust Region Policy Optimization (TRPO) - This algorithm allows more precise control on the expected policy improvement than TNPG through the introduction of a surrogate loss.

D. *The Option-Critic Architecture [5]*

Options are closed-loop policies for taking action over a period of time. Examples of options include picking up an object, going to lunch, traveling to a distant city. Options enable temporally abstract knowledge and action to

be included in the reinforcement learning framework in a natural and general way. The option-critic architecture is capable of learning both internal policies and the terminations conditions of options, in tandem with the policy over options, without the need to provide any additional rewards or subgoals. Options help us assemble big targets into multiple sub-goals and use reinforcement learning to solve those subgoals.
The options framework formalizes the idea of temporally extended actions. A Markovian option $\omega \in \Omega$ is a triple (Iω, $\pi\omega$, $\beta\omega$) where:

- I$\omega \subseteq$ S is an initiation set,
- $\pi\omega$ is an intra-option policy and
- $\beta\omega$: S $\rightarrow$ [0,1] is a termination function.

The option value function can be written as:

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

where QU : S x Ω x A$\rightarrow$R is the value of executing an action in the context of a state-option pair.

E. Policy Gradient Methods for Reinforcement Learning with Function Approximation [6]

Function approximation is essential to reinforcement learning. The value-function approach has worked well in many applications, but has several limitations. Rather than approximating a value function and using that to compute a deterministic policy, the paper approximates a stochastic policy directly using an independent function approximator with its own parameters. It also proves that a version of policy iteration with arbitrary differentiable function approximation is convergent to a locally optimal policy with the help of three theorems.

- Policy Gradient
- Policy Gradient with Function Approximation
- Policy Iteration with Function Approximation

F. Learning Options With a Deliberation Cost [7]

This paper discusses the use of deliberation cost to decide how good options should be in the bounded rationality framework. Longer options lead to a better amortization of the deliberation cost and that is key to understanding their benefit in comparison to using only primitive actions. Long options are favored by penalizing frequent option switches. Thus η sets a margin or a baseline for how good an option ought to be: a correction which might be due to approximation error or to reflect some form of uncertainty in the value estimates. By increasing its value, we can reduce the gap in the advantage function, tilting the balance in favor of maintaining an option rather than terminating it. When no deliberation cost is used, termination raises up to 100% very quickly, meaning each option only lasts a single time-step. Using this approach in the option-critic architecture yields both good performance as well as options which are intuitive and do not shrink over time.

G. A Framework For Temporal Abstraction In Reinforcement Learning [8]

MDPs as they are conventionally conceived do not involve temporal abstraction or temporally extended action. They are based on a discrete time step: the unitary action taken at time t affects the state and reward at time t + 1. There is no notion of a course of action persisting over a variable period of time. As a consequence, conventional MDP methods are unable to take advantage of the simplicities and efficiencies sometimes available at higher levels of temporal abstraction.

One of the keys to treating temporal abstraction as a minimal extension of the reinforcement learning framework is to build on the theory of semi-Markov decision processes (SMDPs), which are a special kind of MDP appropriate for modeling continuous-time discrete-event systems. The actions in SMDPs take variable amounts of time and are intended to model temporally-extended courses of action.

SMDP (option-to-option) methods Options are closely related to the actions in a special kind of decision problem known as a semi-Markov decision process, or SMDP

H. High-Dimensional Continuous Control Using Generalized Advantage Estimation [10]

The typical problem formulation in reinforcement learning is to maximize the expected total reward of a policy. A key source of difficulty is the long time delay between actions and their positive or negative effect on rewards; this issue is called the credit assignment problem in the reinforcement learning literature

Value functions offer an elegant solution to the credit assignment problemthey allow us to estimate the goodness of an action before the delayed reward arrives

The paper proposes a family of policy gradient estimators that significantly reduce variance while maintaining a tolerable level of bias. We call this estimation scheme, parameterized by [0, 1] and [0, 1], the generalized advantage estimator (GAE). Related methods have been proposed in the context of online actor-critic methods.

This paper provides justification and intuition for an effective variance reduction scheme for policy gradients, which we call generalized advantage estimation (GAE).

It proposes the use of a trust region optimization method for the value function, which we find is a robust and efficient way to train neural network value functions with thousands of parameters

III. BENCHMARK ALGORITHM

The Proximal Policy Option-Critic (PPOC) algorithm which, just like PPO, works in two stages. In the first stage, the agent collects trajectories of different options and computes the advantage functions using Monte-Carlo returns. Next is the optimization stage where, for K optimizer iterations, we choose M tuples and apply the gradients. We also chose to use a stochastic policy over options, parameterized by an independent vector $\theta\mu$ (as opposed to ϵ-greedy) which we learned under the same policy gradient approach.

for *iteration=1,2,....* **do**
  $c_t \leftarrow 0$
  $s_t \leftarrow s_0$
  Choose o_t with a softmax policy over options $\mu(o_t|s_t)$
  repeat
    Choose a_t according to $\pi(a_t|s_t)$
    Take action a_t in s_t, observe s_{t+1}, r_t
    $\hat{r}_t = r_t - c_t$
    if β *terminates in* s_{t+1} **then**
      choose new o_{t+1} according to softmax $\mu(o_{t+1}|s_{t+1})$
      $c_t = \eta$
    else
      $c_t = 0$
    end
  until *T timesteps*
  Compute the advantage estimates for each timestep;
  for o=o_1,o_2,.... **do**
    $\theta_{old} \leftarrow \theta$
    for *K optimizer iterations with minibatches M* **do**
      $\theta_\pi \leftarrow \theta_\pi + \alpha_{\theta_\pi} \frac{\partial L_t(\theta)^{\text{PPO}}}{\partial \theta_\pi}$
      $\theta_\beta \leftarrow \theta_\beta - \alpha_{\theta_\beta} \frac{\partial \beta(s_t)}{\partial \theta_\beta} (A(s_t, o_t) + \eta)$
      $\theta_\mu \leftarrow \theta_\mu + \alpha_{\theta_\mu} \frac{\partial \log \mu(o_t|s_t)}{\partial \theta_\mu} A(s_t, o_t)$
      $\theta_w \leftarrow \theta_w - \alpha_{\theta_w} \frac{\partial (G_t - Q_\pi(s_t, o_t))^2}{\partial \theta_w}$
    end
  end
end

Proximal Policy Option Critic (PPOC) Algorithm.

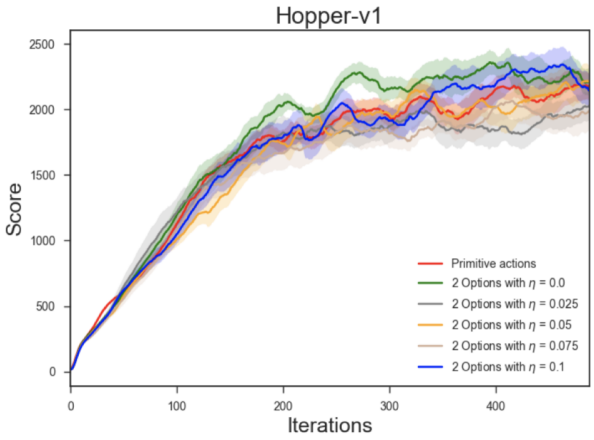

Fig. 2: Result for Hopper

IV. EXTENSION

The Fig. 2 and Fig. 3 represent the results for benchmarks with different deliberation cost(eta values) taken as inputs for the Hopper and the Walker2d environment. One key observation is that an increase in deliberation c=cost does not necessarily guarantee an increase in the performance of the agent. But we can also observe that there exists at least one value of the deliberation cost that performs better while using options when compared with the results for using primitive actions. Hence we could hypothesize that the deliberation cost might depend on the characteristics of the environment. The extension of the algorithm aims at learning the characteristics of deliberation cost parameter by

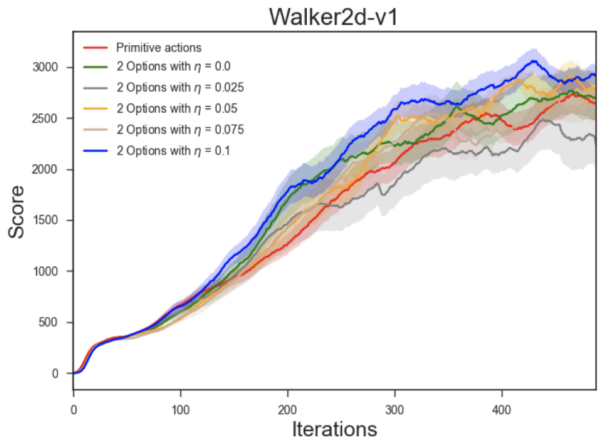

Fig. 3: Result for Walker

running simulations and testing it on various environments. Also, we allow exploration of high deliberation cost values and observe if there is any significant increase in the performance of the agent.

Some characteristics of the deliberation cost:

- Deliberation cost incentivizes the creation of options which persist for a longer period of time. Basically if an agent switches between options frequently, the performance of the agent is worsened. We need an agent policy function that switches between options only when there is a significant increase in the expected reward.
- The option-critic architecture yields both good performance as well as options which are intuitive and do not shrink over time
- Without deliberation cost, the options learn to terminate in each step, hence worsening the performance of the agent.
- An increase in deliberation cost implies a decrease in termination probability
- Our extension aims at learning the characteristics of this η

V. EXPERIMENTAL SETUP

The System used for the simulation was a Linux Machine with Ubuntu 16.04 and 8GB of RAM. Simulations were run on the following 3 MuJoCo environments:

1. Walker-2d:
   - Make a two-dimensional bipedal robot walk forward as fast as possible.
   - Learn deliberation cost(η) that yields the maximum reward.
   - Explore high deliberation cost values to examine performance of agent and the number of iterations it takes to reach the maximum reward.
2. Ant:

- Make a 3D four-legged robot walk
- Learn deliberation cost(η) that yields the maximum reward.
- Explore high deliberation cost values to examine performance of agent and the number of iterations it takes to reach the maximum reward.

3. Amputated Ant:
   - Modified ant environment with one limb amputated.
   - Explore the results when applying the deliberation cost() that yields the maximum reward in the Ant environment.

VI. BENCHMARK IMPLEMENTATION RESULTS

We ran the Proximal Policy Option Critic Algorithm on the Mujoco[11] environment provided by OpenAI gym [10]. This contains various environments that represent different continuous action space learning tasks. We ran the algorithm on 3 tasks,viz., Walker-2d, Half-cheetah and Hopper. We observe the Mean episode length , Mean Episode Reward for 10^6 timesteps.

For all three environments we have provided the deliberation cost(η) = 0.1. Figure 4. represents Reward vs Number of epochs for Walker-2d environment. Figure 5. represents Reward vs Number of epochs for Hopper environment. Figure 6. represents Reward vs Number of epochs for Hopper environment.

Initially the agent tries to perform all possible sequences of actions from an initial state and stores the reward obtained for each action. Once the agent has explored sufficient variety of options(sequences of combination of actions) it tries to exploit those options that give it the maximum expected return. The policy updates are made in small steps to ensure that the learned policy does not deviate too much in terms of quality. The learning is slow because of the small learning rate, small deviation from the previously learned policy. Also the agent has to learn options for multiple joints in order to move without falling.

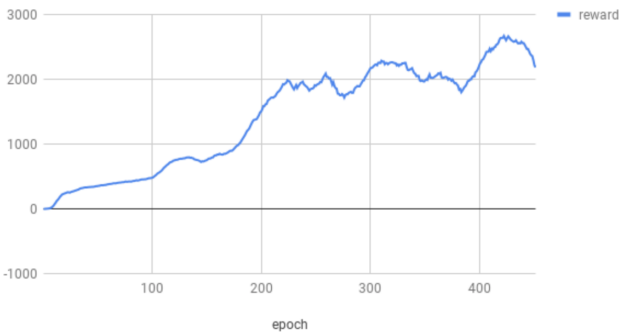

Fig. 4: Walker-2d: Reward vs Number of epochs

VII. EXTENSION RESULTS

- Every agent was allowed to learn for 500 epoch on various η value
- For every environment there are 2 types of result graphs:

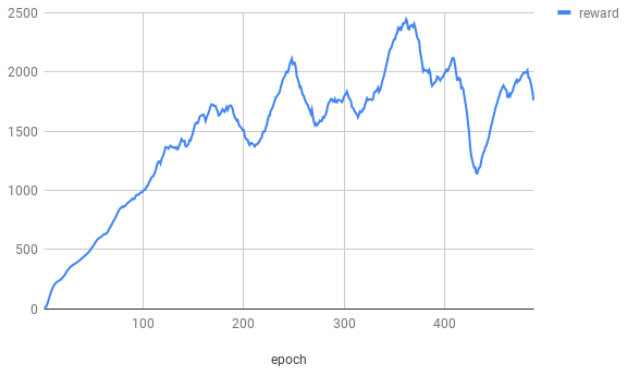

Fig. 5: Hopper: Reward vs Number of epochs

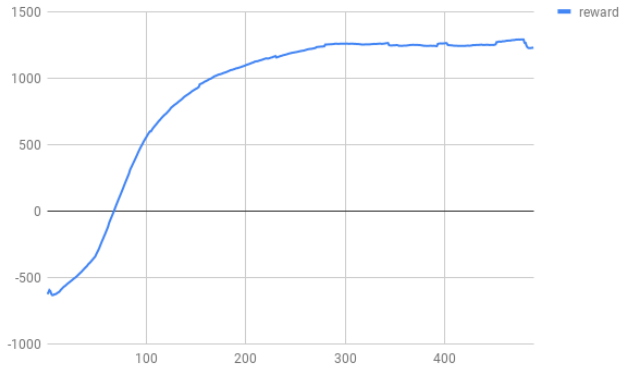

Fig. 6: Half-cheetah: Reward vs Number of epochs

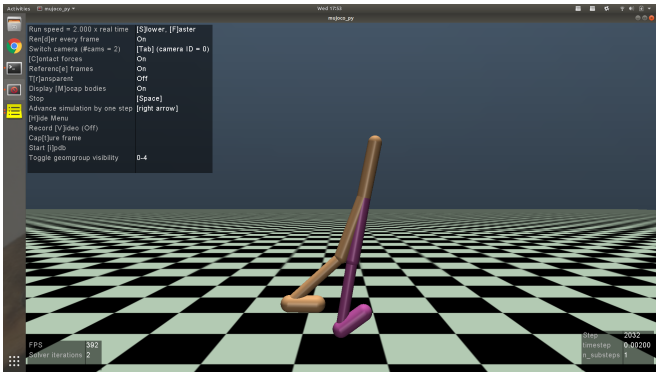

Fig. 7: Training screenshot

- Mean Reward vs No. of Epochs for every environment for the η that yield high rewards
- Max reward value that the agent reaches and the iteration no. when it reaches the reward.

A. *Results for Ant*

If we look at the Ant environment figure, the ant agent has 4 legs which imply 8 limbs. Now to achieve a good reward the agent has to learn optimal options for all these 8 limbs. Partially learning options for some limbs still might cause the agent to get a low reward since it fails to move forward. Hence Ant is a low reward environment which we

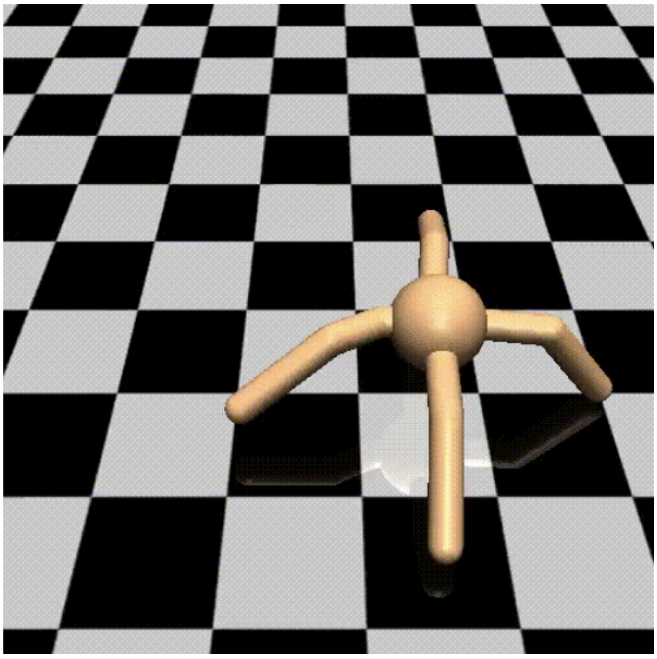

Fig. 8: Environment for Ant

can observe from the plot of Max reward vs epochs. The reward for the ant environment is capped at a value 200. Also looking at the deliberation cost values that give maximum rewards, we observe that, high deliberation cost values actually perform worse on this environment. Also using primitive actions(deliberation cost = 0) the agent performs badly. Deliberation cost of 0.03 ends up being the best value of deliberation cost and has a steady learning curve compared to all other values.

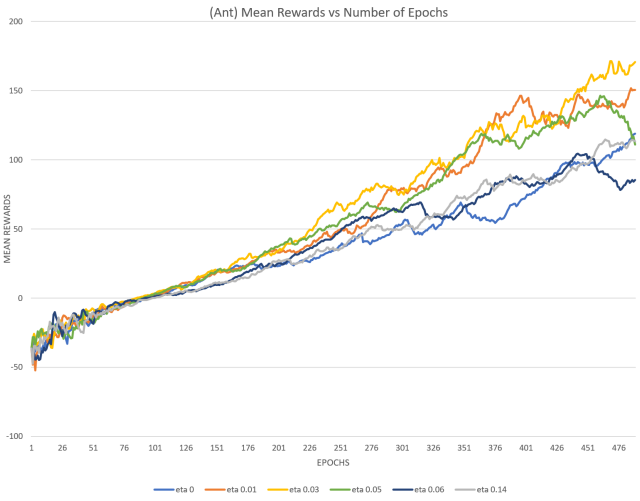

Fig. 9: Results for Ant

Also we can take a look at the graph for the number of iterations taken to reach max reward vs the maximum reward it reaches. This graph clearly indicates that higher values of deliberation cost do not achieve very high reward values and

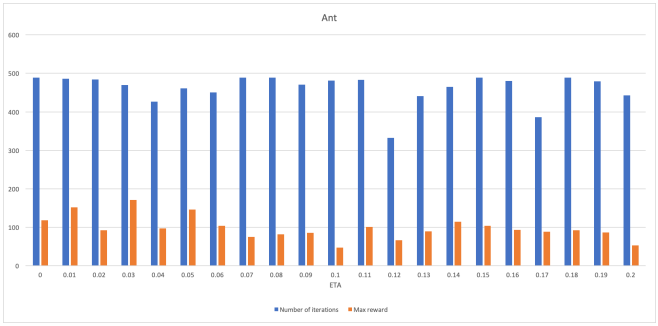

Fig. 10: Rewards and Iteration for reaching max reward per epoch for Ant

take longer to even get to their respective maximum values. Hence deliberation cost of 0.03 was chosen to be applied to the amputated ant environment to test the performance in a manipulated environment with some similar characteristics.

B. Results for Walker

Walker 2d environment differs from the Ant environment in a lot of aspects.

- Firstly, it has only 2 legs which imply 4 limbs. Hence, a lesser number of limbs to learn the optimal policy for. Hence the reward is capped significantly higher than the Ant environment at 400.
- Also, the one key aspect of this environment is that the agent has to learn to stabilize itself whenever it switches between option. The large dips in the graph for Max rewards vs no of rewards represent the attempts of the agent to stabilize itself whenever it tries to switch options between epochs.

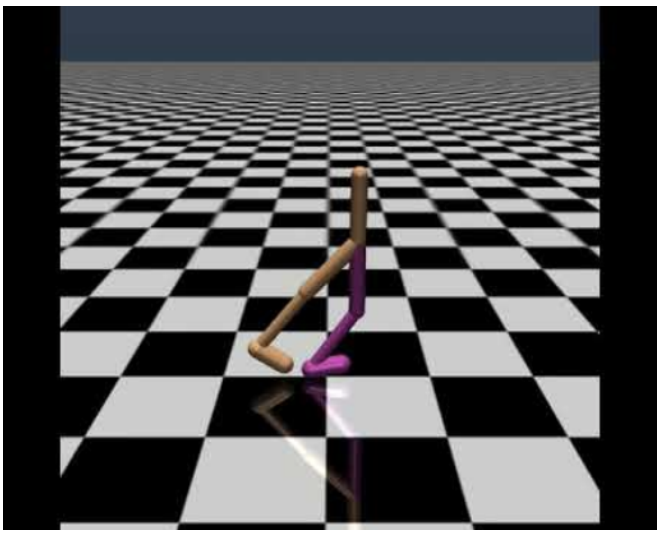

Fig. 11: Environment for Walker

- If we look at the plot for deliberation cost = 0.16 which yields the maximum reward overall, we will observe a large dip around epochs 340-370, where it tries to

switch between option, but the learning becomes fairly stable after that stage.

- In the case of the walker environment, the higher values of deliberation cost perform better.
- If we look at the graph for No of iterations to reach max reward vs Max reward value, we will observe that deliberation cost values above 0.1 yield maximum reward of at least 250 with good consistency.

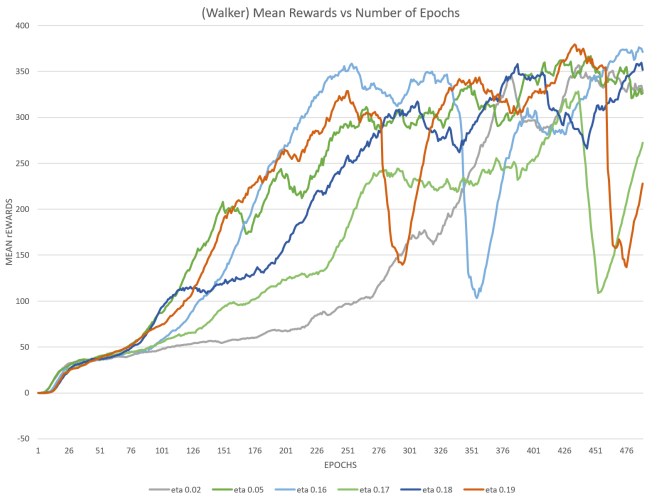

Fig. 12: Results for Walker

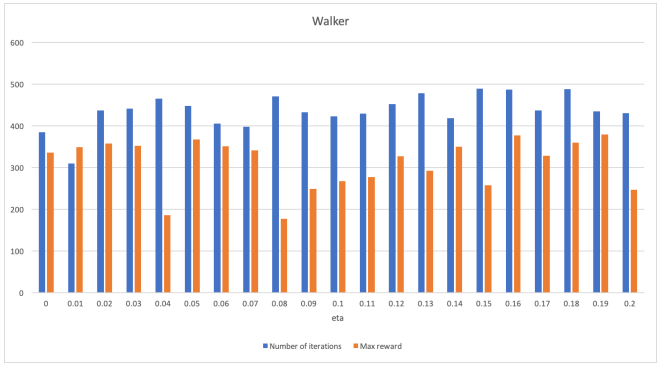

Fig. 13: Rewards and Iteration for reaching max reward per epoch for Walker

C. Applying max reward yielding η on Amputated ant environment

A single limb for the Ant environment agent was amputated and the resulting modified environment can be shown in this Fig. 13

An observation of the learning process of the modified agent can be described as follows:

- Initially, with choice of random options, the Amputated Ant agent topples over and orients itself in an upside-down position.

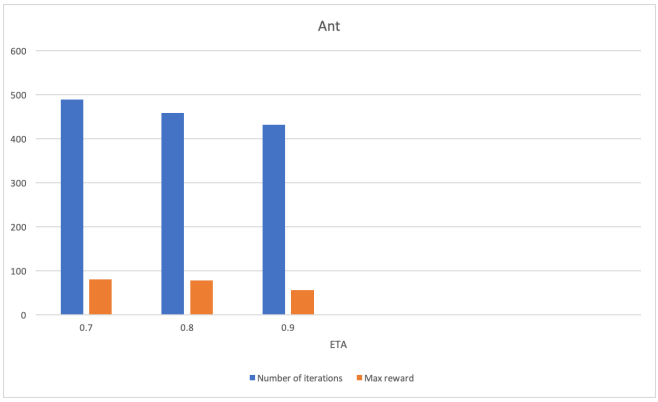

Fig. 14: High dc results for ant

Fig. 15: High dc results for walker

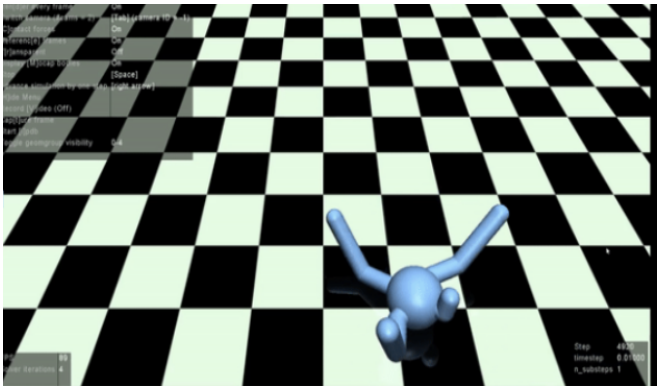

Fig. 16: Amputated ant

- Further, this agent does not try to roll over and stand up.
- The agent rather tries to cover as much ground as it can while being in a flipped state.
- Hence, the agent sticks to a sub-optimal policy which results in it performing worse even when the suitable deliberation cost value is supplied to it.
- This can be clearly inferred for the graph (figure) for Max reward vs No of epoch comparing Ant environment to Amputated ant environment.

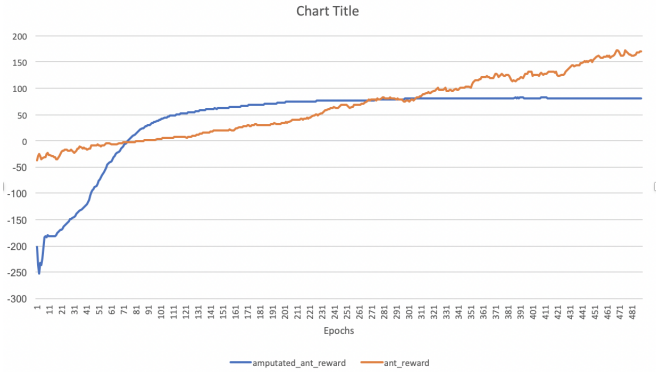

Fig. 17: Reward comparison

VIII. CONCLUSION

- There exists at least one η value using which PPOC performs better than the using primitive actions
- For high reward environments like Walker-2d, deliberation cost (η) value higher than 0.1 yields better performance (Best η value = 0.16)
- For low reward environments like Ant, deliberation cost (η) value lower than 0.1 yields better performance (Best η value = 0.03)
- For higher deliberation cost(η) values, the performance of the agent degrades and it takes a higher number of iterations to reach the maximum reward.

REFERENCES

[1] **Base paper**: Klissarov, M., Bacon, P., Harb, J., Precup, D. (2017). Learnings Options End-to-End for Continuous Action Tasks. ArXiv.org, ArXiv.org, Nov 30, 2017.

[2] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. CoRR, abs/1502.05477, 2015.

[3] Konda, V., Tsitsiklis, J. (2003). OnActor-Critic Algorithms. SIAM Journal on Control and Optimization, 42(4), 1143-1166.

[4] Duan, Y., Chen, X., Schulman, J., Abbeel, P. (2016). Benchmarking Deep Reinforcement Learning for Continuous Control. ArXiv.org, ArXiv.org, May 27, 2016.

[5] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In AAAI, pages 17261734, 2017.

[6] Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In NIPS, pages 10571063, 1999.

[7] Jean Harb, Pierre-Luc Bacon, Martin Klissarov, and Doina Precup. When waiting is not an option : Learning options with a deliberation cost. In AAAI, pages 17261734, 2018.

[8] John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High- dimensional continuous control using generalized advantage estimation. CoRR, abs/1506.02438, 2015.

[9] Richard S. Sutton, Doina Precup, and Satinder P. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. Artif. Intell., 112(1-2):181211, 1999.

[10] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W. (2016). OpenAI Gym. ArXiv.org, ArXiv.org, Jun 5, 2016.

[11] Todorov, E., Erez, T., and Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, pages 50265033. IEEE.