Artificial Intelligence and Robotics

"La Sapienza" University of Rome
Faculty of Information Engineering, Information Technology and Statistics
Department of Informatics, Automation and Control Engineering
"ANTONIO RUBERTI"

SAPIENZA
UNIVERSITÀ DI ROMA

Proximal Policy Optimization Algorithm on Open-AI gym Hopper-v3 Environment

SANNI Oluwatoyin Yetunde[1871835]
Github Link - https://github.com/oluwayetty/hopper_ppo

13.01.2020

1.0 Introduction

This study shows the implementation of Proximal Policy Optimization(PPO) and the model being evaluated on a physical control task (making a hopper take steps without falling) that involve complex multi-joint movements, unstable and rich contact dynamics. For this physical control task, we also compare our results over several experiments.

PPO is a model free member of the family of policy gradient methods for reinforcement learning, which alternate between sampling data through interaction with the environment, and optimizing a "surrogate" objective function using stochastic gradient ascent. Whereas standard policy gradient methods perform one gradient update per data sample, a novel objective function that enables multiple epochs of mini-batch updates is proposed. This involves collecting a small batch of experiences interacting with the environment and using that batch to update its decision-making policy. Once the policy is updated with this batch, the experiences are thrown away and a newer batch is collected with the newly updated policy. This is the reason why it is an “on-policy learning” approach where the experience samples collected are only useful for updating the current policy once.

PPO methods have some of the benefits of trust region policy optimization (TRPO), but they are much simpler to implement, more general, and have better sample complexity (empirically). Although, TRPO has achieved great and consistent high performance, the computation and implementation of it is extremely complicated, the constraint imposed on the surrogate objective function is the KL divergence between the old and the new policy. However, PPO gets rid of the computation created by constrained optimization as it proposes a clipped surrogate objective function. Moreover, PPO uses multiple epochs mini-batches updates instead of performing only one gradient update as per sample like policy gradient methods.

2.0 State of the Art

As earlier stated, the PPO algorithm is mostly equivalent with TRPO algorithm. The difference is that it improves the current state of affairs by introducing an algorithm that attains the data efficiency and reliable performance of TRPO, while using only first-order optimization. Authors of the PPO propose a novel objective with clipped probability ratios, which forms a pessimistic estimate (i.e., lower bound) of the performance of the policy. To optimize policies, they alternate between sampling data from the policy and performing several epochs of optimization on the sampled data.

2.1 Policy Gradient methods

Development of the PPO is built on basic policy gradient methods. Policy gradient methods work by computing an estimator of the policy gradient and plugging it into a stochastic gradient ascent algorithm. The objective function or Policy gradient loss of PG Methods is defined as:

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t\left[\log \pi_\theta(a_t \mid s_t)\hat{A}_t\right]. \qquad [1]$$

- $\widehat{E}_t$ is the expectation which indicates the empirical average over a nite batch of samples.

- $\log \pi_\theta(a_t|s_t)$ is the log probability of the output of the network policy, i.e taking an action a_t at a state s_t
- $\widehat{A}_t$ is the advantage estimate of the relative value of the selected action, if the value is > 0, then this action is better than the other action possible at that state.

Through differentiating equation[1], we obtain the policy gradient estimator which is then plugged into the Stochastic GD algorithm. Hence, pushing the agent to take actions that lead to higher rewards and avoid bad actions. However, there is a problem arising from the step size as if its too small, then the training process will be too slow. On the other hand, if its too high, then there will be too much variability in the training between epochs. Whereas PPO improves the stability of the Actor training by limiting the policy update at each training step. This is done through the Clipped Surrogate Function, which constrains the policy change in a desired range.

2.2 Clipped Surrogate Function

Instead of using log as in equation[1] to trace the impacts of actions, the authors introduce the following probability ratio: $r_t(\theta)$

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \qquad [2]$$

which denotes the probability of an action under the current policy divided by the probability of the action under previous policy. Hence $r_t(\theta_{old}) = 1$.

If $r_t(\theta) > 1$, this means the action is more probable in the current policy than the old policy. However, if $0 < r_t(\theta) < 1$, then the action is less probable for current policy than for the old one. The resulting objective function presented in the PPO paper is:

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right] = \hat{\mathbb{E}}_t\left[r_t(\theta)\hat{A}_t\right]. \qquad [3]$$

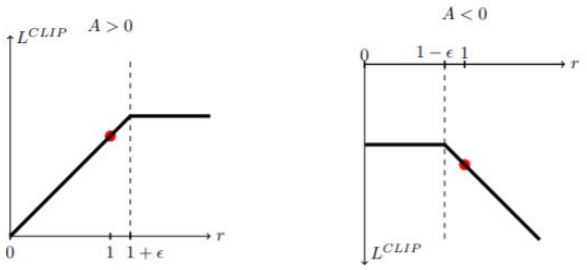
In a situation where the action taken is much more probable in our current policy than in the old one, the lack of a constraint, leads to a large policy gradient step, and a resulting excessive policy update. Hence, there is a need to introduce a constraint which will penalize changes that lead to a ratio that will be far away from 1, which will ensure relatively small policy updates as the new policy can't be too different from the old one. We could use TPRO, which utilizes KL divergence constraints outside of the objective function to constraint the policy update. However, this is as previously stated in the introduction, much more complex and computationally expensive. As a consequence, the author's of PPO introduce a clip probability ratio directly in the objective function, with its clipped surrogate objective function.

$$L_t(\theta) = min(r_t(\theta)\widehat{A}_t,\ clip(r_t(\theta)), 1\text{-}\epsilon, 1\text{+}\epsilon)\widehat{A}_t \qquad [4]$$

- $r_t(\theta)\widehat{A}_t$ is the normal policy gradients objective which pushes the policy towards actions that yield a high positive advantage over the baseline.
- clip($r_t(\theta)$, 1 - ε , 1 + ε) is the clipped version of the normal policy gradients objective.

where epsilon is a hyperparameter,and according to the paper ε = 0.2. Hence, we obtain two probability rations: the first being one non-clipped, and the other clipped in the ε ranges.

Next, we take the minimum of both of these objectives, so the final objective is a lower bound of the unclipped objective. It should be noted that the advantage estimate can be both positive or negative, this is illustrated in Figure 1.



[Fig. 1]

Plots showing one term, of the surrogate function L^{CLIP} as a function of the probability ratio r, for positive advantages (left) and negative advantages (right). The red circle on each plot shows the starting point for the optimization [1]. Figure 1 presents plots of a single term in L^{CLIP}.

In the first case if $\widehat{A}_t$ is greater than 0, it means that the action is better than the average of all the actions in that state. Therefore, the action should be encouraged by increasing $r_t(\theta)$ so that this action has a higher chance to be adopted. Since the denominator of $r_t(\theta)$ is constant, the old policy, increasing $r_t(\theta)$ also implies increasing the new policy $\pi_\theta(a_t|s_t)$. Namely, increase the chance for taking that action in the given state. However, because of the clip, $\pi_\theta(a_t|s_t)$ will only grows to as much as $1+\epsilon$.

By contrast, if $\widehat{A}_t$ is smaller than 0, then that action should be discouraged. As a result, $r_t(\theta)$ should be decreased. Similarly, due to the clip, $r_t(\theta)$ will only decreases to as little as $1-\epsilon$.

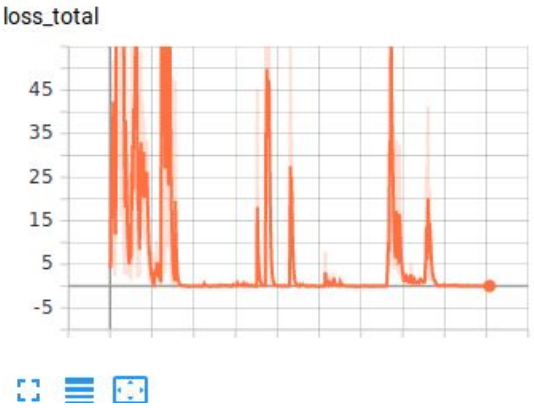
3.0 Experiments and Results

Experiment 3.1 and 3.2 shows the ablation experiments that were performed, their results and observation for the parameters listed in Table 1.

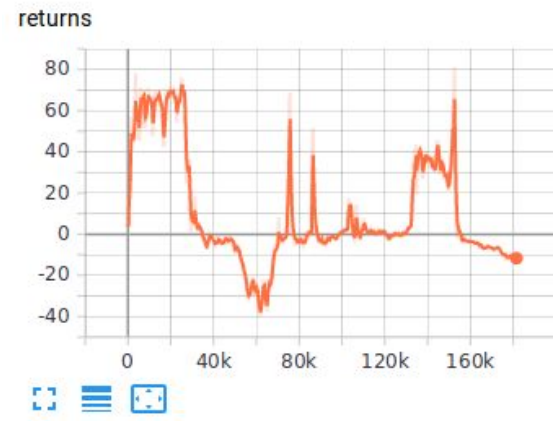
Initial learning rate	0.0003	PPO steps	256
Epsilon (Clipping coefficient for PPO)	0.2	Entropy Scaling coefficient (to encourage exploration)	0.001
Discounting factor	0.99	Mini batch size	64
GAE Lambda	0.95	Critic Discount	0.5
Batch size	128		
Number of environments (agents)	8		
Number of epochs	10		

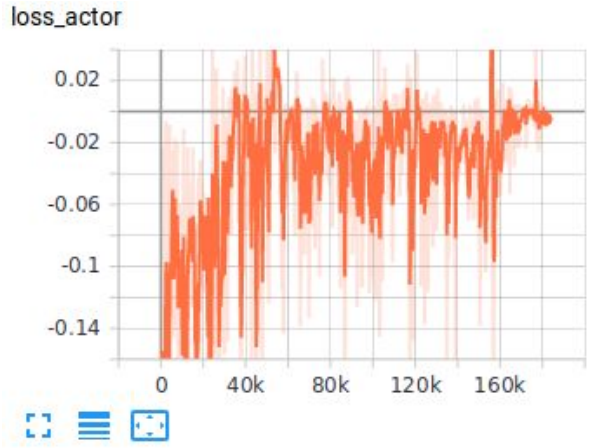
Table 1.0

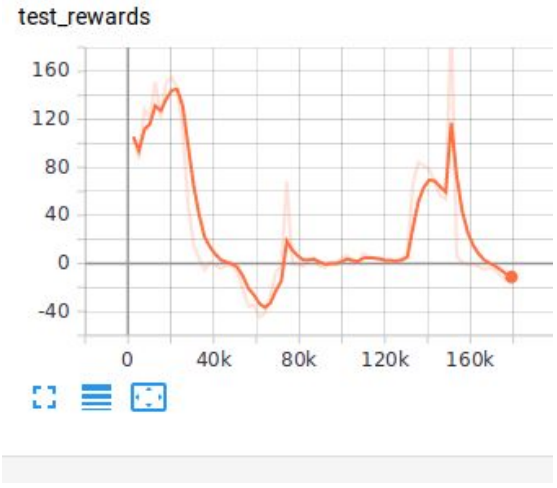
3.1 No clipping/penalty: Using this equation $L_t(\theta) = r_t(\theta)\widehat{A}_t$ for the setting without clipping or penalties, it leads to a very negative score, which is worse.. The figure below shows our observation on its poor performance.

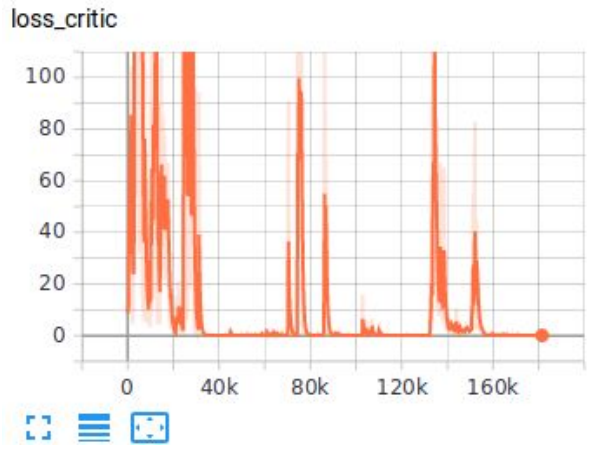



```
Frame 84480. reward: 2.968829089817004
Frame 87040. reward: 4.516522954899532
Frame 89600. reward: -2.457472420911884
Frame 92160. reward: -3.3569060755014006
Frame 94720. reward: 0.858883565647352
Frame 97280. reward: -0.4566503478853969
Frame 99840. reward: 4.329462372063184
Frame 102400. reward: 6.752832163048284
Frame 104960. reward: 1.0719089392717993
Frame 107520. reward: 0.6761256815486405
Frame 110080. reward: 8.393799066875912
Frame 112640. reward: 5.492906719691584
```

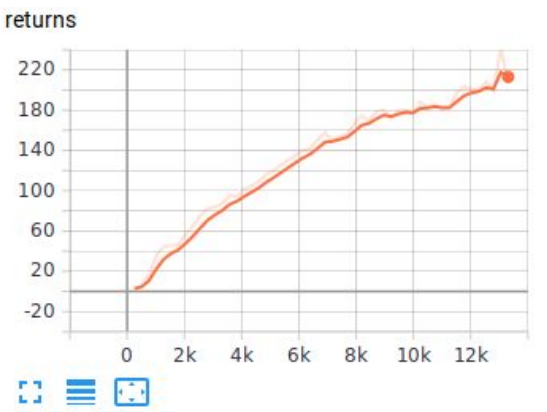


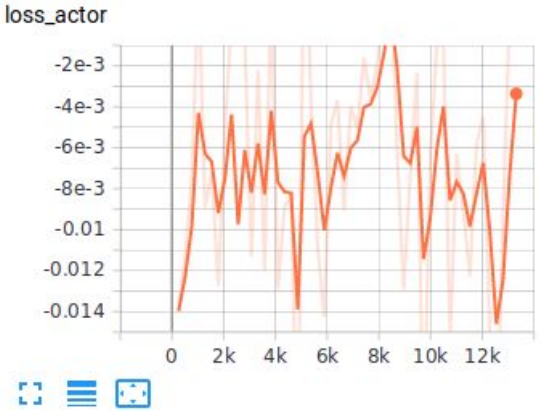






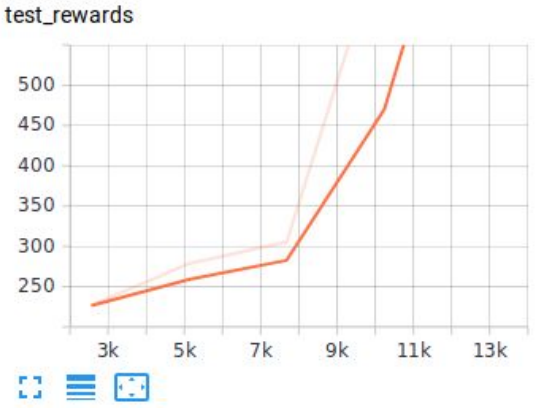
3.2 Clipping: Using this equation $L_t(\theta) = min(r_t(\theta)\widehat{A}_t,\ clip(r_t(\theta)), 1\text{-}\epsilon, 1\text{+}\epsilon)\widehat{A}_t$, which includes the Clipped Surrogate Objective, the following results are shown below.

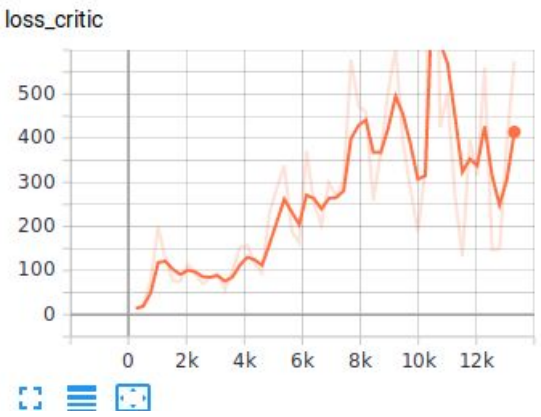


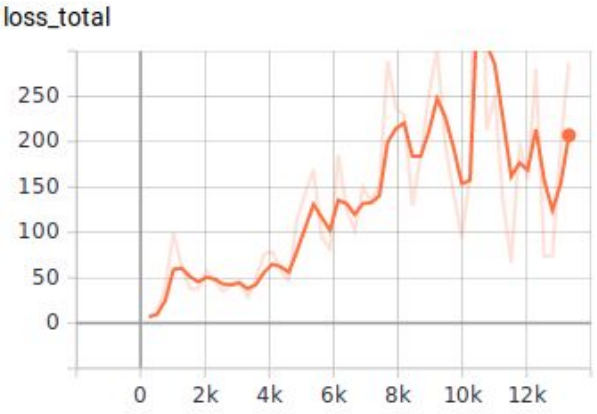


test_rewards

loss_critic

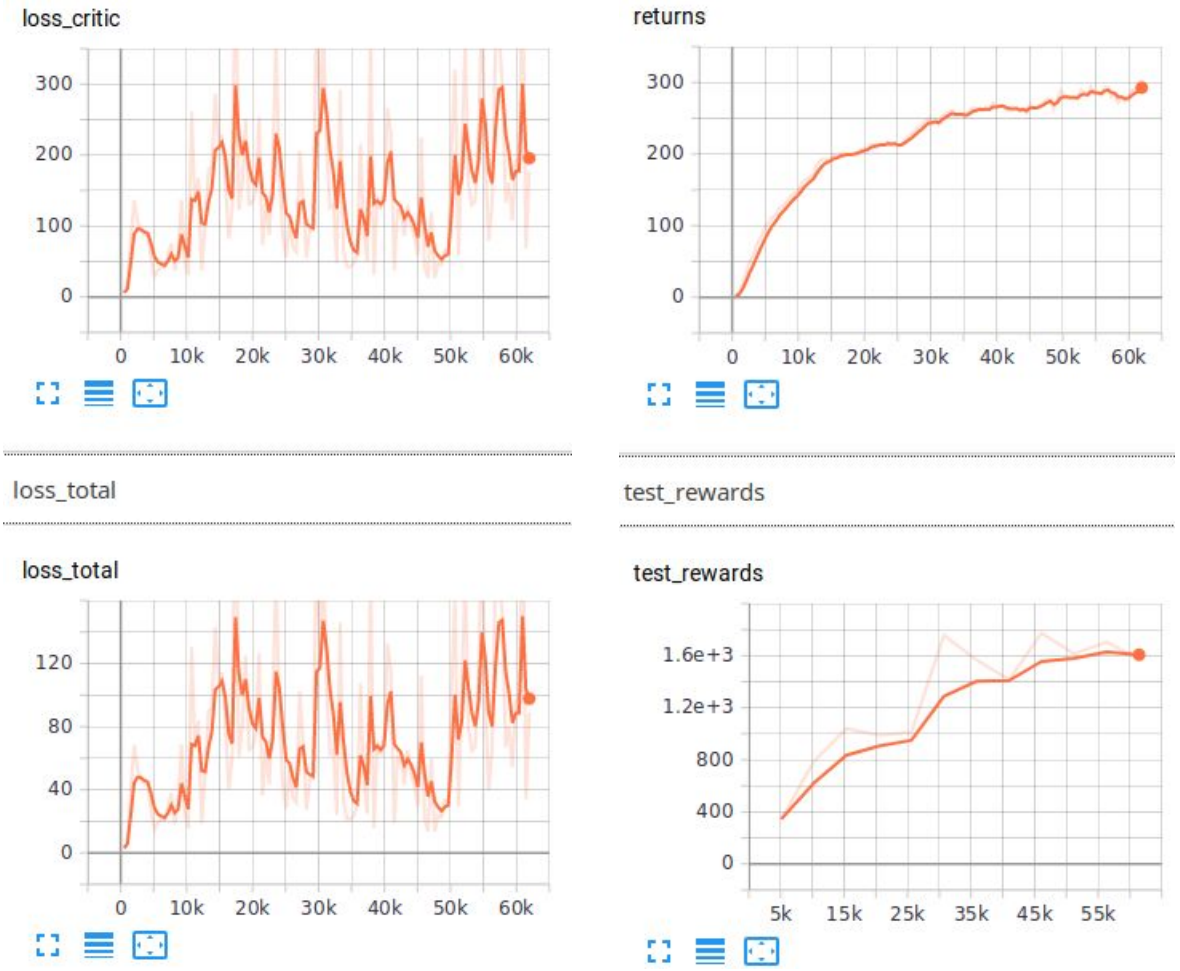






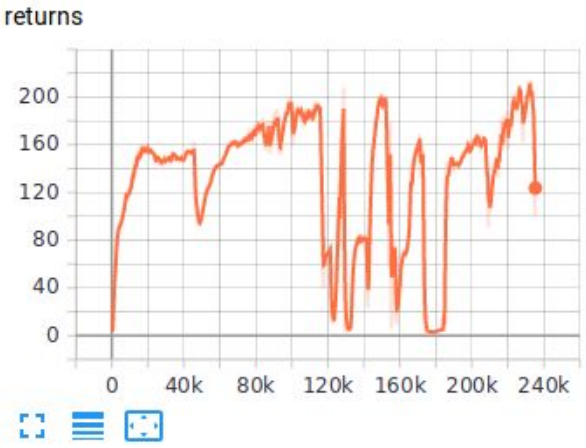
3.3 Clipping with change of different parameters in Table 2: tested The critic loss tends to be bigger than the actor loss, so we scaled it down from **0.5 to 0.3** for another ablation study.

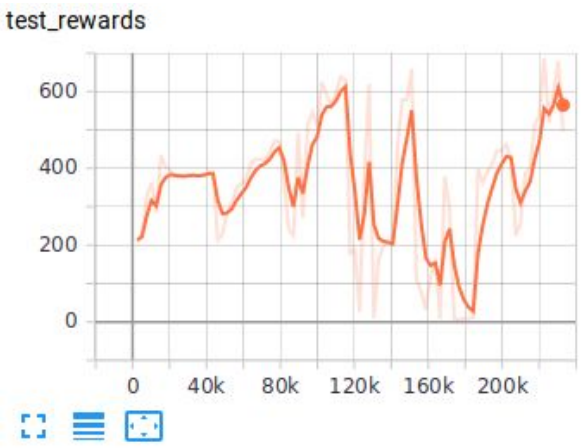
Critic Discount	0.5	Mini batch size	128
Number of environments (agents	16	Entropy Scaling	0.01
PPO_Steps	512	PP0 epochs	15



For this experiment, it will be observed that the actor and critic loss even learns at the same rate, and much better than the configuration settings in 3.1, returns and test rewards also shows a good improvement as the model learns and finally converges.

3.4 Clipping with lower Critic discount and same parameters in Table 1: The critic loss tends to be bigger than the actor loss, so we scaled it down from **0.5 to 0.3** for another ablation study.





We had very unstable spikes in the losses, returns and test rewards, and the training was stopped after a long time to stop because it never converged.

4.0 Conclusion

To conclude, the overall results for using the PPO algorithm with a Clipped Surrogate Objective Function, gave very promising results. However, more training time, on a better workstation with a dedicated GPU is necessary to visualize proper learning and progress. The agent (Hopper-v3) moves gradually by taking very little steps before falling over, which indicates caution and this is probably the result of the clipped range, which prohibits huge policy change. The code can be obtained at this Github repository [6].

The code used can be modified by the addition of a Fixed or Adaptive KL Penalty Coefficient algorithm to the Clipped Surrogate Objective.

References

1. https://blog.openai.com/openai-baselines-ppo/
2. https://mc.ai/summary-proximal-policy-optimizationppo/
3. Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess,Tom Erez, Yuval Tassa, David Silver & Daan Wierstra (2016). Continuous control with deep reinforcement learning. *arXiv:1509.02971v5 [cs.LG] 29 Feb 2016*.
4. Uhlenbeck, George E and Ornstein, Leonard S (1930). On the theory of the brownian motion. *Physical review, 36(5):823*.
5. Watkins, Christopher JCH and Dayan, Peter (1992). Q-learning. *Machine learning. 8(3-4):279–292*.
6. https://spinningup.openai.com/en/latest/algorithms
7. https://github.com/oluwayetty/hopper_ppo