The First International Conference on
Machine Learning and Knowledge Discovery (MLKD 2024)
Amirkabir University of Technology, December 18-19, 2024

Proximal Policy Optimization with Adaptive Generalized Advantage Estimate

Naemeh Mohammadpour* Meysam Fozi† Mohammad Mehdi Ebadzadeh‡ Ali Azimi§ Ali Kamali¶

Abstract

Proximal Policy Optimization (PPO) is one of the leading algorithms in reinforcement learning, designed to optimize policy updates while maintaining stability. However, in complex environments a critical trade-off between bias and variance emerges. Parameter λ in Generalized Advantage Estimate (GAE) plays a crucial role in managing this trade-off, controlling the balance between future and immediate rewards. In this paper, we propose a dynamic adjustment method for parameter λ, based on changes in value loss during training. This adaptive approach enables the model to adjust with respect to variations in the learning process and achieve a better balance between bias and variance. Besides, a policy update delay is introduced to enhance the control of updates in PPO, helping to mitigate large fluctuations in the policy and increase the algorithm's stability. Our experiments show that dynamic λ adjustment significantly improves performance, particularly in complex environments. These results suggest that adaptive λ adjustment is a flexible and effective way to enhance the performance of PPO in various reinforcement learning tasks, especially in challenging and high-dimensional environments, in our case OpenAI Gym Environment Ant-v4 and in DeepMind Control Environment quadrupped walk.

Keywords: Proximal Policy Optimization, Generalized Advantage Estimate, and Bias-Variance Trade-Off

1 Introduction

Reinforcement Learning (RL) has emerged as a powerful tool for solving complex decision-making problems through interaction with environment. Policy gradient methods, [8], which adjust policy parameters to maximize expected rewards, are particularly effective in continuous action spaces, [2, 3, 4, 5].

To address the limitations of traditional policy gradient methods, PPO was introduced as an efficient and robust alternative. PPO improves upon earlier methods by using a clipping mechanism in policy updates, which prevents drastic changes and enhances learning stability. This approach, combined with its computational simplicity, has made PPO one of the most widely used RL algorithms today.

Additionally, techniques such as GAE have been developed to address the bias-variance trade-off in RL, further improving the performance of policy gradient algorithms.

In this paper, we first explain PPO and GAE, then we propose a dynamic adjustment of λ parameter in GAE to achieve a better balance between bias and variance, which is crucial for the stability of the learning process. Also a policy update delay is added to PPO to enhance the control of updates, helping to increase the algorithm's stability. Finally, we evaluate our approach by conducting experiments in two popular environments, OpenAI Gym and DeepMind, to demonstrate the benefits of dynamic parameter tuning on RL performance.

2 Related work

Policy gradient methods optimize policies by directly adjusting parameters to maximize expected rewards. Instead of learning a value function, they update the policy in the direction that increases the probability of rewarding actions. This approach is effective for stochastic policies and continuous action spaces, using function approximators like neural networks.

Proximal Policy Optimization PPO, [7], Algorithm 1, is a type of policy gradient method designed to optimize policies in reinforcement learning while maintaining stability and preventing large, destructive policy updates.

The central idea of PPO is to maximize a "surrogate objective function", allowing multiple epochs of optimization on the same batch of data. Unlike standard policy gradient methods, PPO introduces a mechanism

*Department of Mechanical Engineering, Amirkabir University of Technology, naemeh.mpr@aut.ac.ir
†Department of Computer Engineering, Amirkabir University of Technology, meysam.fozi@aut.ac.ir
‡Department of Computer Engineering, Amirkabir University of Technology, ebadzadeh@aut.ac.ir
§Department of Mechanical Engineering, Amirkabir University of Technology, aliazimi@aut.ac.ir
¶Department of Mechanical Engineering, Amirkabir University of Technology, alikamalie@aut.ac.ir

to limit the extent to which the policy is updated, using either a clipping method or a Kullback-Leibler (KL) penalty.

The surrogate objective in PPO is represented as:

$$L_{\text{clip}}(\theta) = \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t \right], \tag{1}$$

where $\pi_\theta(a_t|s_t)$ is the new policy, $\pi_{\theta_{\text{old}}}(a_t|s_t)$ is the old policy, $\hat{A}_t$ is the advantage estimate, $\hat{\mathbb{E}}_t$ is the empirical expectation over time-steps.

This objective aims to encourage the policy to increase the probability of actions that are deemed advantageous by $\hat{A}_t$.

PPO avoids large, destabilizing policy updates by clipping the probability ratio $\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$. The clipped objective is:

$$L_{\text{clip}}(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t) \right], \tag{2}$$

where:

- $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio.
- ϵ is the clipping hyper-parameter, typically set by a number in the interval $[0.1, 0.3]$. The clip function ensures that if the probability ratio, $r_t(\theta)$, deviates too far from 1 (either below $1-\epsilon$ or above $1+\epsilon$), it gets clipped. This mechanism prevents the policy from changing too drastically in one update, thereby stabilizing training.
- $r_t(\theta)\hat{A}_t$ is the normal policy gradient.
- $\text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t$ limits the update size.
- The use of the min function ensures that the objective is only reduced when the change would lead to an excessively large update, keeping the training stable. This combination of clipping and multiple mini-batch updates leads to efficient and robust policy learning, striking a balance between performance and stability.

Generalized Advantage Estimate. The $\hat{A}_t$ is calculated using GAE, [6], which helps balance the bias-variance trade-off in RL. GAE adds flexibility to the way future rewards are accounted for in policy updates. The advantage is calculated using:

$$\hat{A}_t = \sum_{l=0}^{\infty} (\gamma\lambda)^l \, \delta_{t+1}^V. \tag{3}$$

Algorithm 1 PPO, [7]

```
Initialize θ for policy network
for each iteration do
    for actor = 1, 2, ..., N do
        Run policy π_θold for T timesteps
        Compute advantage estimates Â_1, ..., Â_T
    end for
    Optimize surrogate L wrt θ, with K epochs and
minibatch size M ≤ NT
    θ_old ← θ
end for
```

where γ is the discount factor, which controls the weight given to future rewards, λ is a hyper-parameter, which adjusts the balance between bias and variance, δ_t represents the Temporal Difference (TD) error, which measures the difference between the value of a state and predicted value of the next state. The TD error is calculated as:

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t), \tag{4}$$

where:

- r_t is the reward at time step t.
- $V(s_t)$ is the value estimate for the current state.
- $V(s_{t+1})$ is the value estimate for the next state.

The structure of GAE is similar to $TD(\lambda)$, a method used for value function estimation. However, $TD(\lambda)$ focuses on estimating the value function, while GAE is designed to estimate the advantage function, which helps in policy optimization.

Parameter λ plays a crucial role in managing the bias-variance trade-off in PPO:

- **High λ values (close to 1)**: This setting puts more weight on future rewards, reducing bias but increasing variance. In this case, the advantage estimate sums many future TD errors.

$$\hat{A}_t \approx \sum_{l=0}^{\infty} \gamma^l \delta_{t+1} \tag{5}$$

- **Low λ values (close to 0)**: This setting focuses more on immediate rewards, reducing variance but increasing bias. In this case, the advantage estimate mainly reflects the current TD error.

$$\hat{A}_t \approx \delta_t \tag{6}$$

The parameters γ and λ both help in managing the bias-variance trade-off when working with an approximate

value function. However, they serve distinct purposes and are optimal in different value ranges. The discount factor γ mainly influences the scaling of the value function V^{π}, and operates independently of λ. Setting $\gamma < 1$ introduces bias in the policy gradient estimate, regardless of how accurate the value function is. In contrast, $\lambda < 1$ result in bias only if the value function is not accurate.

3 Proposed methods

PPO with Adaptive GAE. To enhance the performance of PPO, we dynamically adjust λ based on changes in value loss during training. Adjusting λ helps the model adapt to variations in the learning process and a better Bias-Variance trade-off.

When λ is high (close to 1), the advantage estimate, $\hat{A}_t$, heavily depends on future advantages. In this scenario, the network relies more on predictions of future rewards, which can introduce noise and uncertainty, especially if the critic network is not properly trained to provide accurate value predictions.

Algorithm 2 PPO with adaptive GAE

Initialize $v_{\text{loss}}^{\text{old}} \leftarrow \infty$
Initialize total time steps T and global step $\leftarrow 0$
Initialize $\lambda \leftarrow 0.95$ for GAE
for each iteration **do**
  for each episode **do** ▷ Dynamically adjust λ
    Compute value loss v_{loss}
    $m \leftarrow 0.09 - (0.085 \cdot \text{global step}/T)$
    if $v_{\text{loss}} < v_{\text{loss}}^{\text{old}}$ **then**
      $\lambda \leftarrow \max(0.5, \lambda - m)$
    else if $v_{\text{loss}} > v_{\text{loss}}^{\text{old}}$ **then**
      $\lambda \leftarrow \min(0.99, \lambda + m)$
    end if
    $v_{\text{loss}}^{\text{old}} \leftarrow v_{\text{loss}}$
  end for
end for

In the provided implementation, Algorithm 2, λ is adaptively adjusted based on value loss:

- **When value loss decreases**, indicating the critic network is becoming more accurate, λ is reduced, which results in a lower variance. This reduces reliance on future advantages and focuses more on current information, i.e., immediate TD errors.
- **When value loss increases**, indicating poor performance of the critic network, λ is increased, leading to higher variance that can help to improve the algorithm.

The parameter m controls the adjustment rate, initially set to 0.09 and decreased over time. Additionally, this adjustment of λ is more intense at the beginning and gradually decreases over time, allowing the algorithm to have sufficient opportunity to test different λ values with larger steps initially. This approach ensures that the algorithm can explore a wider range of values early on before refining its choices. This approach dynamically tunes λ, aiming to optimize GAE depending on how well the value function is being learned, making it more adaptive to environment's needs. Parameter $\lambda = 0.95$ has been adapted from the original PPO paper, and parameter m is set to linear decay with respect to total timestep. This leads to adaptation to training progress, ensuring more efficient exploration in early stages and refined updates as training stabilizes.

Policy Update Delay (PUD). The concept of delaying policy updates after several updates to the critic was introduced in Twin Delayed Deep Deterministic (TD3) [1] to reduce approximation errors and improve learning stability. This concept can also be applied to enhance the control of updates in PPO, Algorithm 3, helping to mitigate large fluctuations in the policy and increase the algorithm's stability. In our approach, considering the dependency of λ update on the value loss, it is recommended to perform policy updates after several updates to the critic to ensure higher accuracy in learning. Frequent updates to the critic networks compared to the actor provide the critic with more opportunities to minimize value errors effectively.

Algorithm 3 PPO with adaptive GAE and PUD

Initialize policy_frequency $\leftarrow 2$
for each iteration **do**
  for each epoch **do** ▷ Policy Update Delay
    if epoch % policy_frequency == 0 **then**
      Update actor policy L_{actor}
    end if
  end for
  Update critic policy L_{critic}
end for

Complexity Analysis. The overall complexity is primarily driven by factors such as the number of parallel environments, the number of iterations, updates, and the storage of input data (observations), output data (actions), and other information such as values, rewards, and logprobs. In our approach, in addition to the previously mentioned data, the dynamic changes in λ require storing the updated value of this parameter at each update. As a result, the overall complexity of our method remains nearly identical to PPO, and the increase in execution time is negligible.

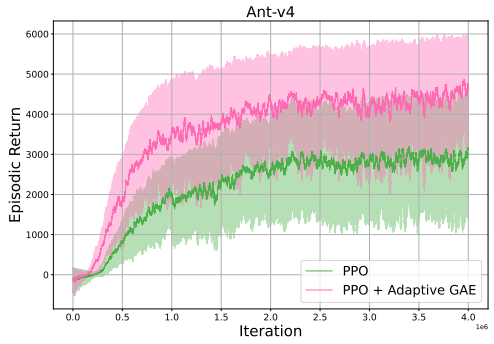

Figure 1: OpenAI Gym `Ant-v4` episodic return in PPO with Adaptive GAE

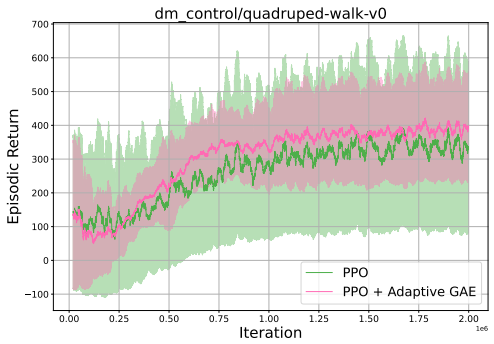

Figure 2: DeepMind `quadruped-walk-v0` episodic return in PPO with Adaptive GAE

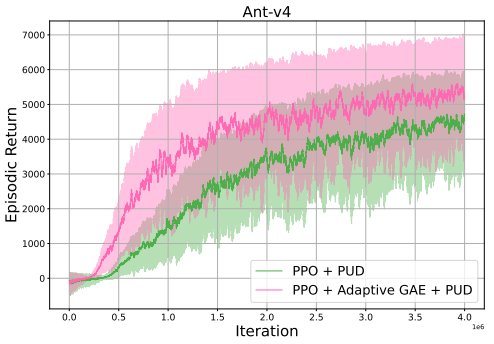

Figure 3: OpenAI Gym `Ant-v4` episodic return in PPO with Adaptive GAE and PUD

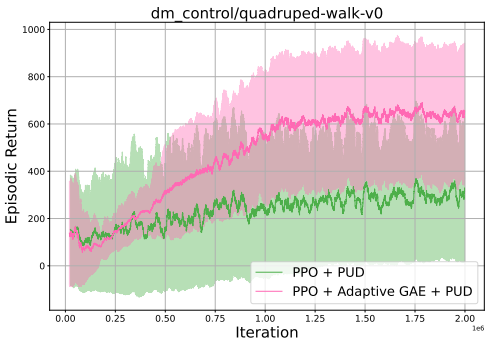

Figure 4: DeepMind `quadruped-walk-v0` episodic return in PPO with Adaptive GAE and PUD

4 Results

We evaluated the proposed methods in two different benchmarks, each with 5 seeds and compared them with PPO as the baseline.

PPO with adaptive GAE. Results on OpenAI Gym MuJoCo Environment, [9], as shown in Figure 1, in four million iterations of the `Ant-v4` benchmark, mean episodic return increased from 3000 to 4500 by keeping variance in a constant range. Results on DeepMind Control Environment, [10], as shown in Figure 2, over two million iterations of the `quadruped-walk-v0` benchmark, mean episodic return increased from 350 to 400, while the variance was significantly reduced.

PPO with Adaptive GAE and PUD. In OpenAI Gym MuJoCo environment, during four million iterations of `Ant-v4` benchmark, we first introduced the policy update delay to PPO. Then, we incorporated PPO with adaptive GAE. Comparing Figure 1 and Figure 3, the policy update delay alone was able to improve PPO (`PPO` vs. `PPO+PUD`). However, when combined with adaptive GAE, as shown in Figure 3, mean episodic return increased from 4500 to 5400 while maintaining the same variance (`PPO + PUD` vs. `PPO + Adaptive GAE + PUD`). The combination of these two methods resulted in beneficial outcomes.

In DeepMind Control environment, during two million iterations of `quadruped-walk-v0` benchmark, we first introduced policy update delay to PPO (`PPO + PUD`). Then, we took adaptive GAE into account (`PPO + PUD + Adaptive GAE`). Comparing Figure 2 and Figure 4, the policy update delay alone resulted in weaker performance compared to PPO (`PPO` vs. `PPO + PUD`). However, when combined with adaptive GAE, it leads to clearly improved results over PPO (`PPO` vs. `PPO + Adaptive GAE + PUD`). Additionally, both the variance and mean episodic return increased; so the combination of both methods does not show significant progress compared to PPO with adaptive GAE (`PPO + Adaptive GAE` vs. `PPO + Adaptive GAE + PUD`). Therefore, policy update delay can, in some cases, lead to improved performance.

5 Conclusion

Based on the provided explanations and the experimental results, we conclude that adaptive adjustment of λ, instead of using a fixed value, can be beneficial and achieve a better balance between bias and variance. This is because, in the early stages of training, when the environment is not properly trained, assigning a higher weight to future rewards may introduce greater uncertainty. The idea of policy update delay, was also introduced and added to the proposed method in order to reduce approximation errors. Rapid policy updates in PPO, which typically result from frequent updates, can lead to issues such as instability. As observed, delayed updates in policy alone can be beneficial in certain benchmarks (e.g., `Ant-v4`). Additionally, due to the dependence of λ on value loss, PUD with adaptive GAE in some cases can help achieve more stable learning and better results.

References

[1] S. Fujimoto, H. Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[2] N. Heess, G. Wayne, D. Silver, T. Lillicrap, T. Erez, and Y. Tassa. Learning continuous control policies by stochastic value gradients. *Advances in neural information processing systems*, 28, 2015.

[3] T. Lillicrap. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[5] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. Trust region policy optimization, 2017.

[6] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[7] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[8] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[9] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[10] S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, N. Heess, and Y. Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020.