The Effects of Decay Schedules on Outer PPO

Abhinav Arya, Tarun Bandi, Sahil Parikh
CMU CS Department, CMU CS Department, CMU Statistics Department
avi1@cmu.edu, tarunbandi@cmu.edu, snparikh@andrew.cmu.edu

Abstract

Complex decision-making tasks have been revolutionized with the advancement in machine learning, with reinforcement learning emerging as a key paradigm for teaching agents to solve dynamic, goal-oriented problems. Among these, Proximal Policy Optimzation (PPO) is a widely-used on-policy reinforcement learning algorithm known for its robustness and adaptability. This paper explores how outer-loop learning rate adjustments and decay mechanisms affect PPO's performance in goal-oriented tasks. Building on recent advancements in decomposing PPO into inner-loop update vector estimation and outer-loop optimization, we implement a variant of PPO, referred to as outer-PPO, that incorporates adaptive learning rates and decay in its outer loop. Experimental results demonstrate that the addition of decay scheduling enhances policy convergence and task performance compared to outer-PPO. These findings underscore the importance of leveraging outer-loop optimization strategies for complex machine learning tasks requiring precision and adaptability.

1 Introduction

Machine learning has transformed the way we approach dynamic decision-making problems, enabling agents to learn optimal strategies in complex and uncertain environments. Reinforcement learning (RL) has been very impactful in the field of learning how to make decisions in situations where decisions need to be made in uncertainty. Common Reinforcement Learning models include value-based methods like Q-learning and Deep Q-Networks (DQN), which approximate the value of state-action pairs to guide decisions. While effective in discrete action spaces, these models struggle in continuous control tasks due to the complexity of maintaining accurate value estimates and their susceptibility to overestimation errors. Policy-based methods, such as REINFORCE, directly optimize policies but often suffer from high variance in gradient estimates, leading to unstable learning. Actor-critic models attempt to combine the strengths of value-based and policy-based approaches but can face challenges with balancing the actor and critic updates, often resulting in suboptimal performance. These drawbacks highlight the need for reinforcement learning algorithms that are both robust and efficient. Proximal Policy Optimization (PPO) addresses many of these issues by introducing mechanisms like clipping and adaptive learning rates, providing a stable and scalable alternative for training agents in dynamic, continuous-action environments.

Proximal Policy Optimization (PPO)[6] is a reinforcement learning algorithm designed to optimize policies efficiently and reliably by addressing issues such as destructive updates and poor sample efficiency that hinder earlier policy gradient methods. At its core, PPO alternates between collecting trajectory data and optimizing a surrogate objective function over multiple minibatches. The key innovation lies in its clipped surrogate objective, which prevents excessively large policy updates by constraining the probability ratio between the new and old policies to remain within a defined interval. This mechanism ensures stable learning while maintaining the flexibility of gradient-based optimization. Additionally, PPO incorporates advantage estimation and entropy bonuses, which help improve exploration and reward prediction during training. In the context of our soccer training task, PPO's ability to handle continuous action spaces and complex reward structures makes it

particularly suitable for teaching a MuJoCo 2D walker to execute precise ball-kicking actions. By leveraging PPO's stability and adaptability, we can train the walker to achieve targeted goals efficiently, demonstrating the algorithm's potential in dynamic, goal-driven reinforcement learning environments.

Building upon the foundation of Proximal Policy Optimization (PPO), our work explores the potential of outer-loop learning rate adjustment and decay mechanisms to enhance policy optimization. Outer-PPO [5] an improvement to standard PPO, decouples the optimization of update vectors in the inner loop from their application in the outer loop, enabling the use of arbitrary gradient-based optimizers. This decomposition allows greater flexibility in adjusting learning rates and introducing momentum-based strategies, addressing inherent limitations in PPO's default unity learning rate. By applying outer learning rates and decay, we can finely tune the magnitude of updates, balancing the trade-off between exploration and stability in dynamic reinforcement learning environments.

In this work, we extend the principles of outer-PPO by emphasizing learning rate decay as a means of attenuating the outer-loop updates over time. This refinement is particularly useful for tasks requiring precise control and gradual convergence, such as training a MuJoCo Humanoid to walk. The decay mechanism allows the policy to adapt more conservatively as it approaches optimal behavior, reducing the risk of overshooting and improving overall stability. By integrating these innovations into PPO, we aim to demonstrate their effectiveness in improving convergence rates and task performance in complex, goal-oriented reinforcement learning settings.

2 Summary of Data

The Gymnasium dataset provides a collection of Reinforcement Learning environments, with MuJoCo (Multi-Joint dynamics with Contact) acting as a Physics backend.

2.1 Capabilities

The environment allows for large details which provide:

1. Efficient contact handling for complex, multi-joint dynamics
2. Fast and accurate rigid body physics
3. Robust numerical integration for stable simulations

2.2 Key Environments

Some powerful environments which are used by MuJoCo include:

Humanoid-v4. A simulator of a full human body which has 17 joints and 24 degrees of freedom. The agent's action space includes complex joint movements, and the agent must learn to walk and coordinate complex movements, counteracting the forces of gravity and conserving momentum.

Action space: Box(-0.4, 0.4, (17,), float32)

An action represents a torque applied on hinges.

Observation Space: Box(-inf, inf, (348,), float64)

The observation space consists of positions, velocity, angles, inertia, constraint force, exertion, and angular velocities.

The total reward is: healthy-reward + forward-reward - ctrl-cost - contact-cost

The reward is defined as the reward for the model being healthy, forward movement, while penalizing large movements, and too much force. Essentially the data aims to make the model make a correct movement

Reward:

The reward is defined as: healthy-reward + forward-reward - control-cost. This would be positive if the model moves forward and stays healthy (in an upright position). Control cost is a regularizer that avoids jerky large movements at any point.

Walker2d. A 2d bipedal walker which must learn forward locomotion, maintain vertical orientation, balance forces across 6 joints, and maximize forward velocity.

Action space: Box(-1.0, 1.0, (6,), float32)

An action represents a torque applied on hinges.

Observation Space: Box(-inf, inf, (17,), float64)

The observation space consists of positions, velocity, angles, and angular velocities.

Reward:

The reward is defined as: healthy-reward + forward-reward - control-cost. This would be positive if the model moves forward and stays healthy (in an upright position). Control cost is a regularizer that avoids jerky large movements at any point.

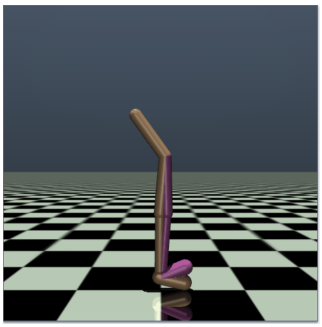

Figure 1: Walker Model

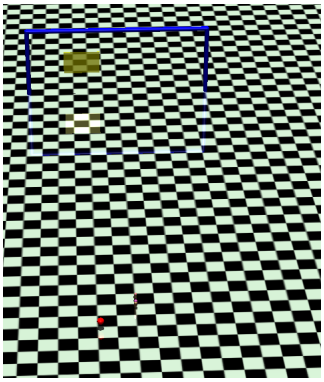

Figure 2: Custom Environment

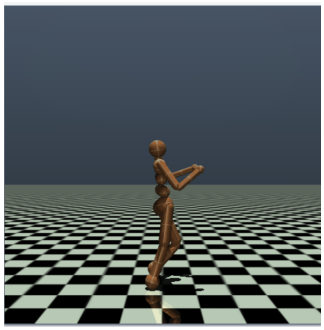

Figure 3: Humanoid Model

These base models are used as benchmarks for most tasks.

We also built our own environment on top of both the humanoid and walker models.

This environment now consists of one of the two above agents alongside a ball and a goal.

For each model, we keep the same observation and action space as before. We simply just add the ball's position and the goal for distance.

The model is now trained to stand and make movements but is awarded by:

Ball proximity, Joint velocity, Goal Scoring, Distance To goal

3 Methods

3.1 Base Reinforcement Learning

Reinforcement learning (RL) provides a framework for solving sequential decision-making problems, where an agent interacts with an environment to maximize cumulative rewards. These problems are modeled as a Markov Decision Process (MDP) defined by (S, A, P, R, γ), where S is the state space, A is the action space, $P(s'|s, a)$ represents transition probabilities, $R(s, a)$ is the reward function, and $\gamma \in [0, 1]$ is the discount factor. The goal is to learn a policy $\pi(a|s)$, which maximizes the expected cumulative reward $J(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$. Key to RL are the value function $V^\pi(s)$, which evaluates the return from a state, and the action-value function $Q^\pi(s, a)$, which evaluates the return from taking action a in state s. These functions underpin RL methods, including policy-based approaches that directly optimize $\pi(a|s)$ and value-based methods that derive policies from $Q^\pi(s, a)$. The policy gradient theorem enables scalable optimization in high-dimensional spaces, forming the foundation of policy optimization methods.

3.2 RL Improvement: Proximal Policy Optimization

Policy Gradient Methods compute estimates of the policy gradient and use it with a gradient descent/ascent algorithm. The common estimator is typically:

$$\hat{g} = \mathbb{E}_t[\nabla_\theta \log \pi_\theta(a_t|s_t)\hat{A}_t]$$

In this case: $\pi_\theta(a_t|s_t)$ is a policy, which is parametrized by θ. $\hat{A}_t$ is an estimator of the advantage function at time step t.

The advantage function can be defined as: $A = Q - V$, where Q is the sum of rewards, and V which is the baseline estimate. We treat the expectation function as an average over many timesteps.

Some methods use constraint optimization for the problem:

$$\begin{aligned} \underset{\theta}{\text{maximize}} \quad & \hat{\mathbb{E}}_t \left[\frac{\pi_\theta\left(a_t \mid s_t\right)}{\pi_{\theta_{\text{old}}}\left(a_t \mid s_t\right)} \hat{A}_t \right] \\ \text{subject to} \quad & \hat{\mathbb{E}}_t \left[\mathrm{KL}\left[\pi_{\theta_{\text{old}}}\left(\cdot \mid s_t\right), \pi_\theta\left(\cdot \mid s_t\right) \right] \right] \leq \delta \end{aligned}$$

This policy constrains it to the region, but CLIP is used to fix

Let $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$

Proximal Policy Optimization uses the objective:

$$L^{\text{clip}} = \mathbb{E}[clip(\min(r_t(\theta)), 1-\epsilon, 1+\epsilon)]$$

With $\epsilon = 0.2$ this has been proven to be a substantial result in the space of training models.

3.3 Proximal Policy Optimization With Outer Learning Rates and Decay

We can see that Proximal Policy Optimization (PPO) revolutionized reinforcement learning by introducing a clipped surrogate objective to constrain policy updates, ensuring stability and preventing destructive changes. However, this approach implicitly couples the direction of the policy update with the magnitude of the update, as determined by the clipping rate ϵ. While ϵ serves to bound the policy's deviation, it also restricts the adaptability of learning, enforcing fixed-sized updates regardless of the task's complexity or optimization landscape.

Outer-PPO addresses this limitation by decoupling the update direction and magnitude, effectively incorporating a gradient-based optimizer into the outer loop of PPO. Formally, consider the PPO surrogate objective function:

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

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio between the current and previous policies, and $\hat{A}_t$ is the estimated advantage function. The clipping mechanism limits the impact of large policy changes by ensuring $r_t(\theta)$ remains within $[1-\epsilon, 1+\epsilon]$.

In Outer-PPO, the clipped surrogate objective is used to determine the *direction* of policy updates, while a separate gradient step modulates the *magnitude* of those updates. Specifically, the policy parameters are updated as follows:

$$\theta_{k+1} = \theta_k + \eta \cdot g_k,$$

where $g_k = \nabla_\theta L^{\text{PPO}}(\theta_k)$ is the gradient of the PPO objective at iteration k, and η is a learning rate determined by an outer-loop optimization strategy. The learning rate η may be adjusted dynamically using decay schedules or adaptive optimization techniques, allowing for finer control over the step size.

This decoupling introduces greater flexibility in adjusting policy updates based on task requirements. The clipping mechanism ϵ defines the allowable deviation range for exploration, while the learning rate η controls the pace of policy convergence. By leveraging dynamic learning rates and decay, Outer-PPO enables more efficient exploration and exploitation, reducing the risk of convergence to suboptimal policies while maintaining stability. This approach is particularly advantageous for tasks requiring precise adjustments, such as those involving continuous control or high-dimensional action spaces.

3.4 Beyond the baseline

Learning rate adjustment has been extensively studied in previous research. Since learning rate is arguably the most critical hyperparameter, its careful selection is essential for model performance. Learning rate decay schedules systematically decrease the learning rate over training iterations to improve model optimization. We define the outer learning rate as σ.

Algorithm 3 Outer-LR PPO Input: $\boldsymbol{\theta}_0$ (parameters), σ (outer learning rate)

for $k = 0, 1, 2, \ldots$ do

$\boldsymbol{g}_k^O \leftarrow \text{PPOITERATION}\ (\boldsymbol{\theta}_k) - \boldsymbol{\theta}_k$

$\boldsymbol{\theta}_{k+1} \leftarrow \boldsymbol{\theta}_k + \sigma \boldsymbol{g}_k^O$

end for [5]

When $\sigma > 1$ the estimate which is made by the gradient is amplified, meaning the model has increased trust in the estimate of the gradient.

The outer learning rate with a value $\sigma < 1$ reflects increased confidence in the original model predictions, as the gradient offers less weight.

As training progresses and the model approaches convergence, smaller steps become more appropriate, suggesting the need to decay the initial outer learning rate. We aim to investigate how implementing learning rate decay could potentially enhance model performance.

4 Results

In our performance evaluation, we employed TensorBoard visualization techniques to systematically compare the performance of three distinct implementations: standard Proximal Policy Optimization (PPO), Outer-PPO, and our proposed Decay-PPO approach. These visualizations provide a multi-dimensional analysis of algorithmic performance across various critical metrics, enabling a nuanced examination of how learning rate decay and outer-loop optimization strategies influence reinforcement learning dynamics. Through carefully generated graphs, we extracted quantitative insights into policy convergence, reward progression, value function stability, and other key performance indicators that illuminate the comparative strengths and limitations of each algorithmic variant.

For each environment, we evaluate three key metrics:

1. Policy Gradient Loss: Measures the stability of policy updates
2. Mean Reward: Quantifies overall task performance
3. KL Divergence: Indicates policy distribution changes during training.
   The Kullback–Leibler (KL) divergence is a measure of statiscial difference between two probability distributions. Mathematically it is defined as:

$$D_{\mathrm{KL}}(P\|Q) = \sum_{x \in \mathcal{X}} P(x) \log \left(\frac{P(x)}{Q(x)} \right).$$

4.1 Humanoid-V4 Dataset

We compare base PPO, PPO with a constant outer learning rate, and PPO with an exponentially decaying outer learning rate.

Metric Comparison: Gradient Loss, Mean Reward, and Approximate KL

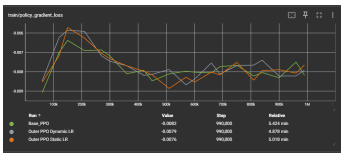

Figure 4: Gradient Loss

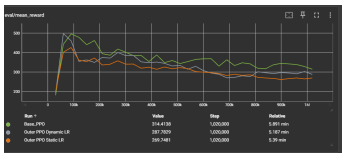

Figure 5: Mean Reward

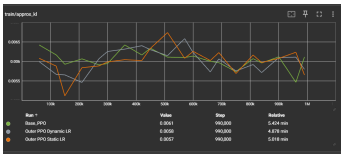

Figure 6: Approximate KL

This comparison shows: - The gradient loss is slightly higher for outer-PPO and decay-PPO compared to base PPO. - Mean reward is highest for base PPO, but decay-PPO outperforms outer-PPO. - Approximate KL is similar across all models, suggesting convergence to similar distributions.

4.2 Humanoid-V4 with Custom Policy

Metric Comparison: Gradient Loss, Mean Reward, and KL Error

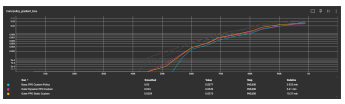

Figure 7: Gradient Loss

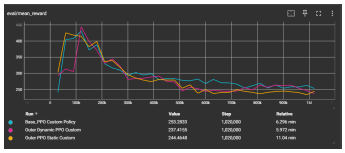

Figure 8: Mean Reward

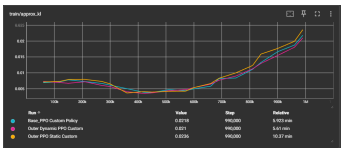

Figure 9: KL Error

The custom policy results indicate: - Gradient loss increases linearly, suggesting optimization adjustments. - Mean reward is slightly better for base PPO compared to the custom policy. - KL error converges similarly across all models, consistent with the clip function's impact.

4.3 Walker-2d Dataset

Metric Comparison: Gradient Loss, Mean Reward, and Approximate KL

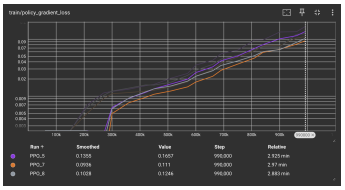

Figure 10: Gradient Loss

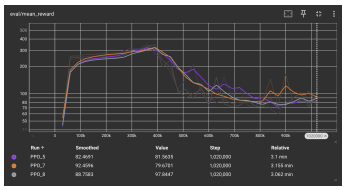

Figure 11: Mean Reward

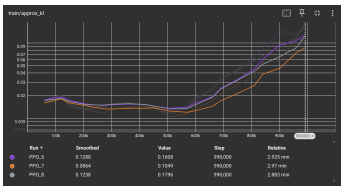

Figure 12: Approximate KL

Walker-2d results reveal: - Gradient loss increases linearly across policies. - Mean reward shows marginal improvement with custom policies. - Approximate KL rises, potentially indicating entrapment in a local maximum.

4.4 Comparative Analysis

Our evaluation reveals that while the decay mechanism improves some metrics in outer-PPO, it still falls short of matching the performance of base PPO. Although our modifications improve performance in specific scenarios, they don't consistently balance exploration and exploitation.

5 Discussion and Analysis

5.1 Reward and Environment Setup for Soccer Agent

The proposed environment for our soccer task aimed to create a comprehensive framework for humanoid agent learning, drawing inspiration from advanced reinforcement learning methodologies.

The design focused on developing a nuanced reward structure that would guide the agent through complex motor control challenges. We envisioned a dense reward mechanism that would break down the soccer task into granular subtasks—positioning, ball interaction, and trajectory optimization. The MuJoCo simulation platform was selected to provide a physics-based environment that could capture the intricacies of continuous motor control. While our initial design emphasized progressive scenario complexity and interpretable action spaces, the practical implementation presented significant technical challenges. The framework sought to balance realistic physical dynamics with actionable learning signals, creating a structured approach to understanding humanoid agent behavior in goal-oriented tasks. Although the fully realized environment remained an aspiration, the theoretical underpinnings provided a robust conceptual foundation for future investigation.

5.2 Hyperparameter Tuning

The selection and calibration of hyperparameters played a critical role in our outer-PPO implementation, with specific parameters demonstrating notable performance implications. The decay rate of 0.99 represents a carefully considered approach to learning rate reduction, capturing a nuanced balance between gradual parameter adjustment and maintaining algorithmic plasticity. This relatively conservative decay rate allows for a smooth, incremental reduction in learning rates, which theoretically should provide a steady dampening of parameter updates without abruptly halting the learning process. The fixed sigma value of 1.5 emerged as a particularly consequential hyperparameter in our experimental design. This value represents a compromise between exploration breadth and policy stability. A sigma of 1.5 suggests a moderately aggressive exploration strategy that permits substantial policy variations while maintaining sufficient constraint to prevent catastrophic divergence. The selection reflects an understanding that reinforcement learning algorithms require a delicate balance between exploring novel action spaces and maintaining the integrity of learned policy representations. Our hyperparameter tuning process involved systematic exploration of these parameters, recognizing that their interaction produces complex effects on policy optimization. The 0.99 decay rate, for instance, creates a gentle exponential reduction in learning rates that allows the algorithm to progressively refine its policy without introducing excessive volatility. Similarly, the sigma value of 1.5 provides a statistical "buffer zone" that enables meaningful policy exploration while preventing uncontrolled randomness.

5.3 Limitations

Our implementation of outer-PPO with learning rate decay revealed methodological limitations that merit critical examination. Specifically, the modified outer-PPO model exhibited inferior task performance compared to standard PPO, raising questions about the viability of the decay approach and the balance between innovation and practical effectiveness in reinforcement learning.

The primary issue lies in the disruption of learning dynamics. Decay scheduling destabilizes the exploration-exploitation balance that standard PPO maintains. In PPO, exploration and exploitation are delicately balanced to refine policies progressively. However, the dynamically changing learning rate in the outer loop introduces systematic noise, creating unpredictable variations in policy update step sizes. This instability prevents smooth policy refinement and leads to erratic, counterproductive adjustments.

Furthermore, the decay mechanism hampers effective exploration of the state-action space. Standard PPO ensures consistent exploration patterns, but the variability in learning rates disrupts this process, resulting in fragmented sampling and less coherent learning outcomes. These inconsistencies increase variance in policy updates, causing abrupt and contradictory shifts in policy representation. This undermines the core premise of policy gradient methods: stable, incremental improvement toward optimal strategies.

These challenges highlight the complexities of algorithmic innovation in machine learning. While our research underscores the potential of outer-loop optimization strategies, it also demonstrates the need for modifications to achieve both theoretical and practical success. Our findings serve as a reminder that algorithmic design must balance innovation with robust performance in real-world applications.

6 Conclusion

This study presented an exploration onto decaying learning rates for PPO. Our research aimed to unlock new capabilities for PPO, but it instead revealed insights about the delicate nature of RL optimization.

Contrary to our assumption, introducing an outer learning rate did not yield great performance improvement in models. Environments including Humanoid-v4 and Walker-2d show that Base PPO typically achieved superior performance to outer PPO and decay PPO variants. Visualization data showed similar convergence patterns where it appeared that base PPO was more stable.

Future research directions might explore alternative approaches to adaptive optimization that preserve PPO's inherent stability while enhancing its adaptability to complex learning environments.

Our work reveals the importance of empirical validation into the learning rate adaptation. A linear decay schedule may perhaps not be the best schedule. A beta decay schedule may provide better results, further research is needed. These insights may guide efforts into improving the tradeoffs between stability, adaptability, and performance.

References

[1] Rahimian, P., Van Haaren, J., Abzhanova, T., & Toka, L. (2022) Beyond action valuation: A deep reinforcement learning framework for optimizing player decisions in soccer.

[2] Sanabria, M., Precioso, F., Mattei, P.-A., & Menguy, T. (2022) A multi-stage deep architecture for summary generation of soccer videos. *arXiv preprint arXiv:2205.00694*. Retrieved from https://arxiv.org/abs/2205.00694.

[3] Fernández, J., & Bornn, L. (2021) SoccerMap: A deep learning architecture for visually-interpretable analysis in soccer. In *Machine Learning and Knowledge Discovery in Databases. Applied Data Science and Demo Track*, pp. 491–506. Springer International Publishing. Retrieved from http://dx.doi.org/10.1007/978-3-030-67670-4_30.

[4] Kim, D. (2020) DeepSoccer. *GitHub repository*. Retrieved from https://github.com/kimbring2/DeepSoccer/, commit 9ccab28a7e2a9a14caa119a765f95e2c6d0b044e.

[5] Tan, C. B., Toledo, E., Ellis, B., Foerster, J. N., & Huszár, F. (2024) Beyond the boundaries of proximal policy optimization. *arXiv preprint arXiv:2411.00666*. Retrieved from https://arxiv.org/abs/2411.00666.

[6] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017) Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*. Retrieved from https://arxiv.org/abs/1707.06347.

[7] Humanoid - Gym Documentation. (2022) *Gymlibrary.dev*. Retrieved from https://www.gymlibrary.dev/environments/mujoco/humanoid/.