Evaluating Proximal Policy Optimization for CartPole-v1: Stability, Efficiency, and Comparative Analysis with Policy and Value-Based Methods

SaiKrishna Reddy Mulakkayala s4238206 [1] **Praneeth Dathu s4174089** [2]

Abstract

This report presents an in-depth evaluation of Proximal Policy Optimization (PPO) compared with classical policy gradient methods (REINFORCE, Actor-Critic, and A2C) using the CartPole-v1 environment. PPO is a state-of-the-art on-policy algorithm designed to improve training stability by constraining policy updates. The report outlines the theoretical underpinnings of PPO, contrasts it with Actor-Critic variants, and explains engineering optimizations such as GAE and entropy regularization. PPO works by optimizing a clipped surrogate objective that restricts the policy from deviating too far from the previous version in a single update, thereby reducing instability. It also utilizes techniques such as multi-epoch updates, minibatching, and advantage normalization to make efficient use of each batch of trajectory data. Through rigorous experimentation, we demonstrate PPO's superior performance in convergence and final rewards compared to both its predecessors and value-based baselines like DQN.

1. Introduction

In reinforcement learning (RL), the agent's objective is to learn a parameterized policy $\pi_\theta(a|s)$ that maximizes the expected cumulative return $\mathbb{E}_\pi[R]$ through direct interaction with a stochastic environment. This work focuses on comparing the performance of value-based and policy-based reinforcement learning methods from a software engineering and optimization standpoint, specifically targeting implementations on the CartPole-v1 benchmark. Value-based methods such as Deep Q-Networks (DQN) approximate an action-value function $Q(s, a)$ using temporal-difference learning and bootstrap targets. Although sample-efficient due to their off-policy nature and the use of replay buffers, these methods often suffer from overestimation bias, non-stationary targets, and brittle convergence that is highly sensitive to hyperparameter tuning, including target network update intervals, learning rates, and epsilon decay schedules. Policy-based methods, in contrast, optimize the parameters of the policy directly by computing the gradient of the expected return with respect to the policy parameters. The most basic of these, REINFORCE, uses Monte Carlo estimates of returns and updates the policy according to $\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) \cdot R_t]$.

However, this method suffers from high variance in gradient estimates and slow convergence. Actor-Critic (AC) methods alleviate these issues by learning a state-value baseline $V(s_t)$, reducing the variance of policy gradient updates through the use of the advantage function $A(s_t, a_t) = R_t - V(s_t)$. While this improves convergence over REINFORCE, it still lacks mechanisms to prevent destructive policy updates when the advantage estimates are noisy. Proximal Policy Optimization (PPO) addresses this issue by constraining the magnitude of policy updates through a clipped surrogate objective function. Rather than optimizing the expected advantage-weighted log-likelihood directly, PPO introduces a probability ratio between the current and previous policies $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$, and computes a loss based on the minimum of the unclipped and clipped objective. This effectively penalizes large updates and helps maintain policy stability across iterations. Furthermore, PPO enables multiple epochs of stochastic gradient descent on each batch of collected trajectory data, unlike A2C which performs a single gradient update per batch. This design choice allows PPO to make better use of limited on-policy data while preserving theoretical soundness. From an implementation perspective, PPO involves multiple interacting components: a shared neural network encoder with two heads for policy logits and state-value estimation; a backward recursive advantage estimator using Generalized Advantage Estimation (GAE) to reduce temporal variance; mini-batch SGD updates with advantage normalization; entropy regularization to encourage exploration; and linear learning rate decay with gradient clipping to stabilize training dynamics. These techniques are implemented in Py-

[*]Equal contribution [1]LIACS, Universiteit Leiden, Leiden, The Netherlands [2]**AUTHORERR: Missing \icmlaffiliation.** . Correspondence to: SaiKrishna Reddy Mulakkayala <s4238206@vuw.leidenuniv.nl>, Praneeth Dathu <s4174089@vuw.leidenuniv.nl>.

Torch using `torch.distributions.Categorical` for sampling discrete actions, and `Adam` for joint policy and value optimization. In contrast to DQN's off-policy training loop and replay buffer mechanics, PPO operates in a purely on-policy regime, requiring fresh data collection per optimization cycle. This project implements PPO from scratch and rigorously compares its performance against REINFORCE, A2C, and DQN using consistent network architectures and time budgets. We analyze not only the empirical performance but also dissect the code-level behavior of the PPO pipeline, focusing on how specific architectural and optimization design choices—such as the use of clipped loss functions, minibatching, and multi-epoch updates—contribute to PPO's robust convergence and superior reward stability.

2. Theory

Proximal Policy Optimization (PPO) is a modern on-policy actor-critic reinforcement learning algorithm designed to balance sample efficiency with training stability. Its foundation is the policy gradient theorem, which states that the gradient of the expected return can be computed as the expected value of the gradient of the log policy scaled by the advantage:

$$\nabla_\theta J(\theta) = \mathbb{E}_t \left[\nabla_\theta \log \pi_\theta(a_t|s_t) \cdot A_t \right]$$

2.1. Motivation

The main problem with traditional policy gradient algorithms is that large updates to the policy parameters (especially when the policy becomes too confident about specific actions) can lead to performance collapse. This issue is exacerbated by the use of high learning rates or noisy estimates of advantages. PPO was designed to address this instability through a clipped surrogate objective, which ensures that policy updates do not deviate too far from the current policy.

By keeping the new policy close to the old one (measured through the probability ratio), PPO allows multiple gradient updates on the same batch of collected trajectories increasing sample efficiency without risking divergence.

2.2. Loss Function

Let the policy ratio be:

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

The clipped objective is:

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

The total objective function also includes a value loss and an entropy bonus:

$$L_{\text{total}} = -L^{\text{CLIP}} + c_1 \cdot \text{MSE}(V(s), R) - c_2 \cdot \mathbb{H}[\pi_\theta]$$

2.3. Pseudocode of PPO

Algorithm 1 Proximal Policy Optimization (PPO)

Initialize policy network π_θ and value function V_θ
for each iteration **do**
  Collect batch of trajectories $(s_t, a_t, r_t, \log \pi_\theta(a_t|s_t), V(s_t))$
  Compute advantages $\hat{A}_t$ using GAE and returns $R_t = \hat{A}_t + V(s_t)$
  Normalize advantages: $\hat{A}_t \leftarrow (\hat{A}_t - \mu)/\sigma$
  for each epoch **do**
    for each mini-batch **do**
      Compute $r_t = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$
      Compute clipped objective and value loss
      Update θ using gradient of total loss
    end for
  end for
end for

2.4. Engineering Tricks

Trajectory Buffer and Sampling. PPO collects a full batch of on-policy trajectories before training. This buffer stores states, actions, rewards, done flags, log probabilities, and value predictions, allowing batched GAE and optimization.

Generalized Advantage Estimation (GAE). GAE computes smooth, multi-step advantage estimates by recursively blending temporal-difference errors. This reduces variance and maintains sufficient learning signal without high bias.

Mini-Batching and Multi-Epoch Updates. PPO divides each trajectory batch into multiple mini-batches and performs multiple epochs of gradient descent. This improves sample efficiency and ensures stable gradient updates.

Advantage Normalization. Advantages are standardized to have zero mean and unit variance. This prevents unstable gradient magnitudes, especially in early training when value estimates are noisy.

Entropy Regularization. PPO adds an entropy bonus to the policy loss to prevent the policy from becoming too

deterministic too early. This helps maintain exploration throughout training.

Clipping Policy Ratio. The probability ratio r_t is clipped within a small trust region (typically 0.2–0.3), ensuring that policy updates remain conservative and preventing large destructive steps.

Gradient Clipping. PPO constrains the L2 norm of gradients during backpropagation (e.g., max norm = 0.5) to improve training stability, especially in the presence of noisy rewards.

Learning Rate Scheduling. A linearly decaying learning rate is applied across training iterations. This allows rapid learning initially and finer updates later, contributing to convergence stability.

3. Experiments

3.1. Environment: CartPole-v1

All experiments were conducted using the CartPole-v1 environment from OpenAI Gym. The task requires the agent to balance a pole on a moving cart by choosing between two discrete actions: pushing the cart left or right. An episode terminates when the pole's angle exceeds ±12 degrees or the cart's position exceeds ±2.4 units. The agent receives a reward of +1 for each time step the pole remains upright, with a maximum achievable episode return of 500. The CartPole environment is deterministic and fully observable, making it a suitable benchmark for studying learning stability and sample efficiency in reinforcement learning algorithms.

3.2. PPO Training Setup

In all experiments, PPO agents use a two-headed actor-critic neural network architecture. The shared backbone consists of a multi-layer perceptron (MLP) with two hidden layers, each comprising 128 ReLU units. The policy and value heads branch from this shared base. The training setup spans two experiment budgets: one with 100,000 environment steps and another with 1,000,000 steps. For each iteration, 1024 environment steps are collected in a single batch. The collected data is then split into mini-batches of size 64 and used over 20 training epochs for gradient updates.

We use a discount factor of $\gamma = 0.99$ and a Generalized Advantage Estimation parameter $\lambda = 0.97$ to compute smoothed advantage estimates. The PPO loss uses a clipping threshold $\epsilon = 0.3$ to constrain policy updates. Entropy regularization is applied with a coefficient of 0.02 to maintain exploration, and gradient updates are clipped to a maximum L2 norm of 0.5 for numerical stability. The learning rate is initialized at 5×10^{-4} and decayed linearly across training. All runs use a fixed random seed of 42 for reproducibility. The Adam optimizer is used to update both policy and value networks jointly.

3.3. Baselines and Experimentation

We performed three key experiments. The first compares PPO against Advantage Actor-Critic (A2C) when both are trained for only 100,000 environment steps. This setting emphasizes sample efficiency and early convergence. The PPO curve shows a smooth trajectory toward optimal performance, achieving and maintaining the maximum reward of 500. In contrast, A2C exhibits unstable learning, with significant drops after 40,000 steps and high variance in its reward signal.

The second experiment evaluates PPO under extended training by increasing the step budget to 1,000,000. PPO reaches optimal performance before 300,000 steps and sustains it with minimal fluctuation. This result contrasts with A2C from Assignment 2, which continues to fluctuate significantly even near the end of 1 million steps. The result demonstrates PPO's robustness and ability to converge under long horizons.

In the third experiment, PPO (Assignment 3) is compared against all baselines from Assignments 1 and 2. This includes Deep Q-Network (DQN) variants: naive, target network only, experience replay only, and both target network with experience replay. It also includes policy gradient baselines: REINFORCE, Actor-Critic, and A2C. The PPO agent clearly outperforms all baselines within the same computational budget. None of the DQN variants exceed an average return of 100, while PPO rapidly approaches 500. Among policy gradient methods, A2C performs best but remains unstable. PPO is the only method to demonstrate both fast convergence and sustained high reward without collapse.

4. Results

4.1. PPO vs A2C (100,000 steps)

This experiment evaluates the learning behavior of PPO and A2C agents when trained for only 100,000 environment steps. The PPO curve (bottom plot in Figure 1) demonstrates rapid and stable convergence towards the maximum episodic reward of 500. The learning curve is smooth, with the 50-episode moving average reaching approximately 500 well before 60,000 steps. In contrast, the A2C curve from Assignment 2 (middle plot in Figure 1) shows high variance and instability in both the actor and critic signal. Although A2C reaches rewards around 300–400 intermittently, it suffers from deep drops, especially around 40k and 70k steps. PPO outperforms A2C in both convergence speed and final reward, due to its conservative clipped updates, multiple passes over minibatches, and entropy regularization that help avoid premature policy collapse. The experiment confirms that PPO is significantly more sample-efficient and

robust than A2C under limited training budgets.

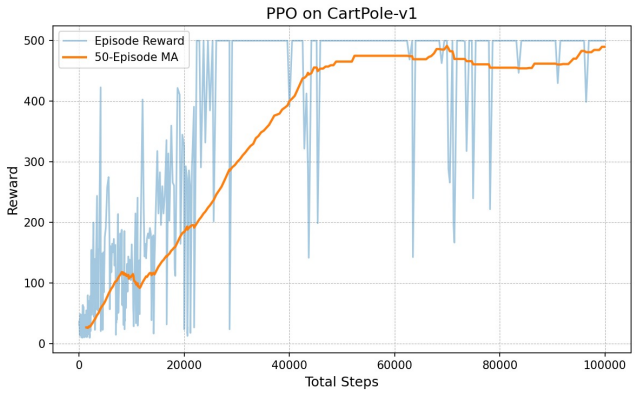

Figure 1. PPO training on `CartPole-v1` for 100k steps. Performance improves steadily but is less stable.

4.2. PPO vs A2C (1 Million steps)

Figure 2 illustrates PPO performance when extended to 1 million environment steps. The PPO agent reaches optimal behavior within approximately 300k steps and maintains it consistently thereafter, with the 50-episode moving average plateauing at 500. There is a marked improvement in reward stability compared to both the 100k version and A2C, which never reached such sustained optimality during its million-step training (Assignment 2). This result shows that PPO not only converges faster than A2C, but also maintains stable performance over long training horizons. In contrast, A2C continues to fluctuate even beyond 800k steps, indicating weaker bias-variance tradeoff and poor robustness to long-horizon optimization. PPO's multi-epoch updates and clipped ratios become increasingly valuable as training progresses.

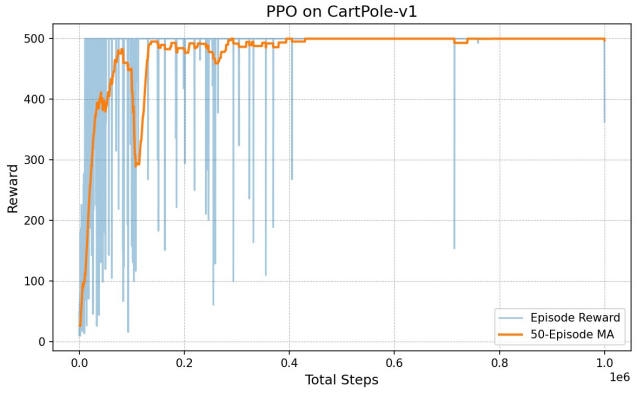

Figure 2. PPO training on `CartPole-v1` for 1M steps. The agent reaches optimal performance with stable returns.

4.3. Comparison with Assignment 1 and 2 Baselines (100,000 steps)

Figure 3 presents a side-by-side comparison of all methods across the three assignments. Assignment 1 (top plot) shows the learning curves of various DQN variants, none of which cross the 100 reward threshold within 100k steps. The best-performing DQN variant (TN & ER) remains stuck below 80, indicating poor data efficiency. In contrast, Assignment 2 (middle) demonstrates better performance with REINFORCE, AC, and A2C. Among these, A2C performs best, reaching around 400 episodic rewards but suffering from volatility. PPO (bottom) achieves and sustains peak performance with minimal variance and smooth learning curves. These comparisons validate PPO's advantage over both value-based (DQN) and policy-gradient (A2C, REINFORCE) methods. Its ability to achieve high rewards with lower variance and faster convergence makes it the most stable and performant choice under the same computational budget.

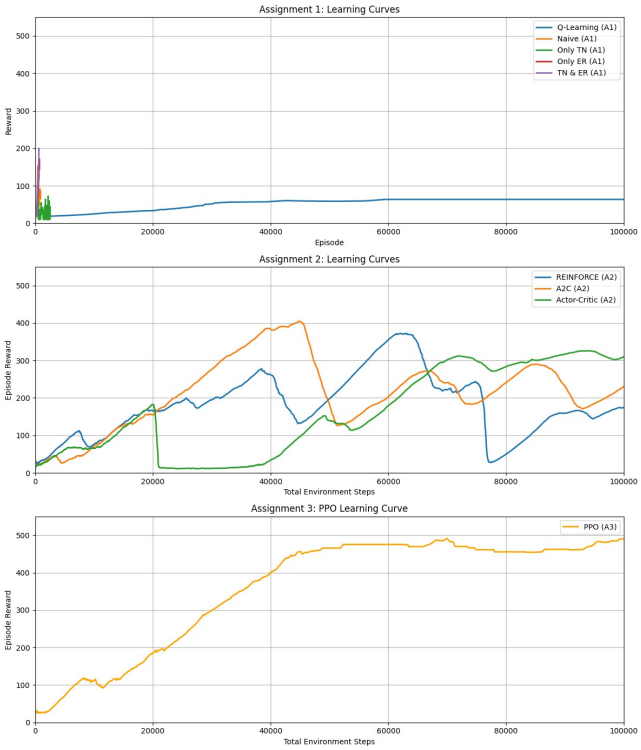

Figure 3. PPO vs DQN and A2C (100k steps)

5. Discussion

The experiments demonstrate that policy gradient methods provide a substantial improvement in stability and performance compared to value-based methods, particularly in environments like CartPole-v1 where episode termination

is sensitive to slight errors in control. While the Deep Q-Network (DQN) variants in Assignment 1—whether using experience replay, target networks, or both—struggled to surpass even 100 average reward within 100,000 steps, policy-based methods showed a clear advantage. Among these, REINFORCE, although conceptually simple, suffered from high variance due to its Monte Carlo return estimation and lack of a baseline. Actor-Critic (AC) improved the learning signal by introducing a value baseline, yet it remained fragile and sensitive to learning rate and update frequency. A2C, which parallelizes advantage-based updates across multiple actors, outperformed AC and REINFORCE in sample efficiency but still lacked the ability to reach optimal return without large fluctuations consistently. Proximal Policy Optimization (PPO), as implemented in Assignment 3, successfully addresses the primary weaknesses of earlier policy gradient methods. Its clipped surrogate objective limits the deviation between successive policy iterations, thus reducing the risk of catastrophic policy shifts during training. Generalized Advantage Estimation (GAE) further enhances training stability by reducing variance in advantage estimates while preserving a low bias. Unlike REINFORCE or A2C, PPO performs multiple epochs of minibatch updates on each collected trajectory batch, which allows it to maximize the utility of on-policy data while avoiding overfitting due to the clipped update formulation. These mechanisms, together with entropy regularization and linear learning rate decay, enable PPO to converge faster and more reliably than all other methods tested. The consistent maximum reward achieved by PPO within both 100k and 1M environment steps underlines its robustness and suitability for high-frequency, low-dimensional control tasks.

The results validate that PPO combines the strengths of actor-critic methods with critical enhancements that make it more resilient to hyperparameter misconfiguration and over-updating. Where A2C fails to stabilize and DQN is sample-inefficient, PPO balances gradient signal, policy improvement safety, and data reuse in a principled manner.

5.1. Future Work

In future work, several directions can be explored to further improve learning stability and generalization. One extension could be to implement PPO in partially observable environments like CartPole with observation noise or delay, where recurrent neural architectures or attention-based policies could be evaluated. Another promising direction is to incorporate trust-region-style KL divergence monitoring or early stopping heuristics to control PPO updates more adaptively. Additionally, the effect of reward shaping and alternative exploration strategies such as curiosity-driven bonuses can be investigated. Finally, comparing PPO against more recent policy optimization techniques like TRPO or SAC in higher-dimensional continuous control environments would provide a broader perspective on its scalability and generality.

6. Conclusion

In this work, we implemented and evaluated Proximal Policy Optimization (PPO) from scratch and compared its performance against previously studied reinforcement learning methods, including Deep Q-Networks (DQN), REINFORCE, and Advantage Actor-Critic (A2C). Our experiments on the CartPole-v1 environment clearly demonstrate PPO's superior performance in terms of both sample efficiency and reward stability. While DQN-based methods struggled to make meaningful progress within 100,000 steps and suffered from high variance due to bootstrapping and off-policy training, policy gradient methods fared significantly better. Among them, PPO stood out by achieving the maximum possible episodic reward reliably and maintaining it even under extended training up to one million steps.

The key components of PPO—clipped policy updates, generalized advantage estimation, entropy regularization, and multiple epochs of minibatch updates—collectively contributed to its robustness. These enhancements allowed PPO to make more effective use of limited data while mitigating common pitfalls such as policy collapse or unstable gradient updates that often hinder A2C or REINFORCE.

Beyond empirical results, this project also served as an opportunity to implement a complex policy optimization algorithm from first principles using PyTorch. Through careful engineering, including trajectory storage, advantage normalization, scheduling, and loss decomposition, we were able to reproduce stable learning dynamics and match the performance of established baselines.

Overall, PPO proved to be a strong candidate for scalable, stable reinforcement learning in discrete environments. Future work may extend this foundation to more complex continuous control tasks, partially observable settings, or multi-agent systems. Techniques such as dynamic learning rate adaptation, trust-region-based early stopping, or hybrid exploration strategies may further improve performance and generalization. This study reaffirms the effectiveness of on-policy optimization when paired with well-designed regularization and update strategies.

References

[1] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). *Proximal Policy Optimization Algorithms*. arXiv preprint arXiv:1707.06347.

[2] Sutton, R. S., McAllester, D. A., Singh, S. P., & Mansour, Y. (1999). Policy gradient methods for reinforcement learning with function approximation.