Assignment 3: Advanced Actor Critic- PPO and/or SAC

Kirthik Raja Kesavan(s4570170) [1] **Fernando Fejzulla(s4534719)** [1] **Ekansh Khanulia(s4336089)** [1]

Abstract

We study how different improvements affect Proximal Policy Optimization (PPO) on the CartPole-v1 task. We start with simple two-layer policy and value networks. Then, we add four techniques: Generalized Advantage Estimation (GAE), multi-epoch minibatch updates, entropy regularization, and gradient-norm and ratio clipping.With these improvements, PPO reaches the maximum average return of 500 in about 200,000 steps. This is more than twice as fast as vanilla PPO, and three to four times faster than A2C. On the other hand, DQN and REINFORCE do not reach this performance, even after one million steps. Each added method helps the learning process by reducing variance, improving sample use, encouraging exploration, or keeping updates stable. Overall, our improved PPO gives fast and stable results and can be used as a strong baseline for other control tasks.

1. Introduction

Reinforcement learning has progressed from early value-based methods like Deep Q-Networks (DQN)—which approximate action-values but suffer from overestimation bias and are limited to discrete actions—to policy-gradient approaches such as REINFORCE, and then to simple actor–critic hybrids like A2C. In Assignments 1 and 2 we observed that DQN can struggle with stability in continuous environments, while REINFORCE and A2C, despite learning stochastic policies directly, face high variance and can only use each sample once per update.

In this work, we apply Proximal Policy Optimization (PPO) to the CartPole-v1 environment—where an agent must apply horizontal forces to keep a pole balanced upright—to address these shortcomings. Unlike DQN (critic-only) and REINFORCE/A2C (single-pass policy updates), PPO jointly optimizes a policy and value network using a clipped surrogate objective that constrains large updates and reuses data across multiple epochs. To further improve sample efficiency and stability, our implementation incorporates Generalized Advantage Estimation, mini-batch updates , entropy regularization, and gradient clipping . We will evaluate how each of these engineering enhancements affects learning dynamics and compare against the performance from our earlier assignments.

2. Theory

2.1. PPO

PPO is an on-policy actor–critic algorithm designed to improve stability and sample efficiency by limiting policy updates. Each iteration it collects trajectories, computes advantages, and then reuses the same data for multiple epochs of mini-batch gradient updates under a clipped surrogate objective.

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

where $r_t(\theta)$ is the probability ratio between the new and old policies, $\hat{A}_t$ are the estimated advantages, and ϵ controls the trust-region size. A value-function loss is likewise minimized against discounted returns. Clipping $r_t(\theta)$ bounds policy updates, preserving exploration and marrying trust-region safety with first-order simplicity. We chose PPO over SAC because it requires fewer hyperparameters and offers a more straightforward on-policy setup, which aligns well with CartPole-v1's need for stability and rapid convergence. PPO's clipped objective prevents large policy shifts—avoiding the overfitting dips we saw—whereas SAC's off-policy design and dual Q-networks introduce extra complexity and tuning overhead without significant upside in this simple task.

2.2. Policy (Actor) Loss

The actor in our PPO implementation is trained by minimizing the clipped surrogate objective

$$L_\pi(\theta) = -\frac{1}{N}\sum_{t=1}^{N}\min\Big(r_t(\theta)\,\hat{A}_t,\ \text{clip}\big(r_t(\theta), 1-\epsilon, 1+\epsilon\big)\,\hat{A}_t\Big),$$

where $r_t(\theta) = \pi_\theta(a_t \mid s_t)/\pi_{\theta_{\text{old}}}(a_t \mid s_t)$ is the probability ratio between the updated policy and the one that generated the data, and $\hat{A}_t$ is the Generalized Advantage Estimate at

time$t$. The clipping operation $\mathrm{clip}(r, 1-\epsilon, 1+\epsilon)$ enforces a soft trust region by bounding r_t within $[1-\epsilon, 1+\epsilon]$, and the $\min$ operator selects the more conservative of the unclipped and clipped terms—capping both overly large positive updates (when $\hat{A}_t > 0$) and overly large negative updates (when $\hat{A}_t < 0$). The leading negative sign and the average over the batch ensure compatibility with standard gradient-descent optimizers, which seek to minimize the loss. This formulation stabilizes learning by preventing destabilizing policy shifts while still reinforcing actions that yield above-baseline returns and suppressing those that do not.

2.3. Critic Loss

The critic in our PPO implementation is trained by minimizing the mean-squared error between its value predictions and the target returns. Denoting by $V_\phi(s_t)$ the predicted value of state s_t and by R_t the discounted return computed during rollout (i.e. $R_t = r_t + \gamma\, R_{t+1}$ bootstrapped at the end), the critic loss is

$$L_v(\phi) = \frac{1}{N}\sum_{t=1}^{N}\big(V_\phi(s_t) - R_t\big)^2.$$

2.4. Pseudo Code

Algorithm 1 PPO pseudo code

```
Initialize policy π_θ, value V_φ networks, and Adam optimizers
while total_steps less than max_steps do
    Collect T steps: (s_t, a_t, r_t, mask_t, log π_θold, V_φ(s_t))
    Bootstrap V_φ(s_{T+1}); compute returns R_t and advantages Â_t via GAE(λ)
    Form dataset of size N, shuffle and split into minibatches of size B
    for epoch = 1 to K and each minibatch do
        Compute r_t = exp(log π_θ − log π_θold), policy loss (clipped + entropy), and MSE value loss
        Clip gradients and update θ and φ
    end for
end while
```

We initialize actor and critic networks with Adam optimizers and loop until a set number of environment steps. In each iteration, we collect transitions—states, actions, rewards, masks, log-probs, and values—then bootstrap the final value and apply GAE to compute returns and advantages. These transitions are flattened, shuffled, and split into mini-batches for multiple epochs, where we compute the clipped surrogate policy loss (with an entropy bonus) and the MSE critic loss. We apply gradient clipping and update both networks each minibatch. Training ends when the step budget is exhausted.

2.5. Relation to Actor–Critic (A2C) Methods

Equations (2) and (3) lay out the basic policy and critic losses used in reinforcement learning, and we can see how they connect to the PPO setup in the code. Equation (2),

$$\mathcal{L}(\theta) = \mathbb{E}_{s\sim\mathcal{D}}\Big[\,\mathbb{E}_{a\sim\pi_\theta(\,a|s)}\big[-Q^{\pi_\theta}(s,a)\big]\Big].$$

shows the policy loss. It averages over states and actions from the old policy, mixing the new policy's log-probability with the old policy's Q-value. This comes from policy-gradient theory, which pushes the policy toward actions with higher expected returns, but it can be unstable if updates are too large. Equation (3),

$$\mathcal{L}(\theta) = \mathbb{E}_{s\sim\mathcal{D}}\Big[\,\mathbb{E}_{a\sim\pi_\theta(\,a|s)}\big[-A^{\pi_\theta}(s,a)\big]\Big].$$

gives the critic loss, fitting the value function to match the advantage over states from a dataset. This objective reduces variance in policy gradients by providing a good baseline estimate. In the code, the policy loss

$$\mathcal{L}(\theta) = -\mathbb{E}_t\Big[\min\big(r_t(\theta)\,\hat{A}_t,\ \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\,\hat{A}_t\big)\Big]$$

builds on Equation (2) by using the probability ratio

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

and clipping it, a core PPO modification that constrains policy updates. The critic loss

$$\mathcal{L}(\phi) = \mathbb{E}_t\big(V_\phi(s_t) - R_t\big)^2$$

follows Equation (3) but uses bootstrapped returns R_t from GAE rather than raw advantages, smoothing the fitting process. These adjustments make both losses more stable, aligning with PPO's goal of avoiding large swings in learning.

PPO maintains the actor–critic framework—using separate policy and value networks—but replaces the vanilla loss $\mathbb{E}[\log \pi_\theta(a|s)\,\hat{A}]$ with the clipped surrogate

$$-\mathbb{E}_t\big[\min(r_t\,\hat{A}_t,\ \mathrm{clip}(r_t, 1-\epsilon, 1+\epsilon)\,\hat{A}_t)\big],$$

where $r_t = \pi_\theta/\pi_{\theta_{\text{old}}}$. This enforces a trust region without costly second-order methods. PPO also uses GAE ($\lambda = 0.95$) for smoother advantages, whereas A2C relies on simpler TD or Monte-Carlo estimates. The critic still minimizes MSE $\mathbb{E}[(V_\phi(s) - R_t)^2]$, but fits to GAE returns. Finally, PPO's multi-epoch mini-batch updates (5 epochs, batch size 64), gradient clipping (max-norm 0.3), and entropy bonus (0.06) further stabilize learning compared to standard A2C.

3. Experiments

3.1. Experimental Setup

CartPole-v1 is a classic control environment in which an agent observes a four-dimensional continuous state—cart position, cart velocity, pole angle, and pole angular velocity—and selects one of two discrete actions (push left or push right). At each time step the agent receives a reward of +1 for keeping the pole's angle within ±12° of vertical and the cart's position within ±2.4 units of the track center; otherwise the episode terminates. Episodes also end after 500 steps if those limits are never exceeded. Each episode begins with all state variables initialized uniformly at random in [–0.05,0.05], providing slight variation in starting conditions while keeping the task focused on learning balance control. To boost PPO's stability and efficiency, we incorporate four key engineering tricks—Generalized Advantage Estimation, mini-batch optimization, entropy regularization, and gradient clipping—which are explained in detail in the next section

3.2. Engineering Tricks

Generalized Advantage Estimation (GAE) (Schulman et al., 2015) computes the advantage by blending one-step TD errors (reward plus discounted next value minus current value) with longer-horizon returns, weighting earlier errors more heavily via an exponential decay factor . Reason for choosing this is that by setting 0.95, GAE interpolates between the low-variance bias of single-step estimates and the low-bias but high-variance of full Monte-Carlo returns, reducing overall estimation variance without introducing excessive bias. This spectral mix yields more reliable advantage signals, smoothing out noisy reward fluctuations and stabilizing policy updates, which allows PPO to reach near-optimal returns on CartPole-v1 far more quickly—though occasional performance drops still reflect residual instability

Mini-batching (Schulman et al., 2017) breaks the collected rollout of N transitions into smaller subsets (batch size 64) and applies gradient updates over each subset for 5 epochs. Using mini-batches in our implementation rather than the full rollout at once reduces peak memory usage and increases the number of weight updates per sample, improving the precision of gradient estimates and speeding overall convergence. By reshuffling and reusing data across epochs, we also mitigate over-fitting to any particular segment of the rollout. Furthermore, frequent, smaller updates better utilize hardware parallelism and smooth out large swings in gradient norms.Mini-batching helps us drive a smooth ascent in average return—reaching around 400 by 200 K steps—though the inherent stochasticity of small batches can introduce variability, manifesting as the occasional dip in performance.

Entropy Regularization (Schulman et al., 2017)encourages the policy to remain stochastic by adding its Shannon entropy to the training objective. This extra term penalizes overly confident (low-entropy) action distributions, ensuring the agent continues to explore different actions rather than prematurely settling on a narrow set. We choose this beacuse without sufficient exploration, an agent can get stuck in a suboptimal policy—especially in tasks like CartPole-v1 where local optima lurk. By weighting the entropy bonus with a coefficient of 0.06, we strike a balance: early in training it keeps the policy diverse, and later it prevents collapse onto a limited action set. Impact: this moderate entropy coefficient, tuned alongside our learning rates (policy = 1e-4, value = 1e-3) and other tricks, helps the PPO agent discover robust behaviors more reliably while still converging to high rewards.

Gradient Clipping (Pascanu et al., 2013) is a technique that caps the norm of the gradient vector during backpropagation so that no individual update can exceed a specified threshold. By enforcing a maximum gradient norm (in our case, 0.3), we prevent sudden, large parameter jumps that could destabilize learning.we choose Gradient clipping becauase in deep reinforcement learning the high variance of returns and the non-stationarity of the policy can lead to very large gradients, which, if left unchecked, may cause the network weights to diverge. So gradient clipping ensures that every update remains within a controlled range, reducing the risk of divergence and maintaining consistent, stable performance throughout training.

3.3. Results

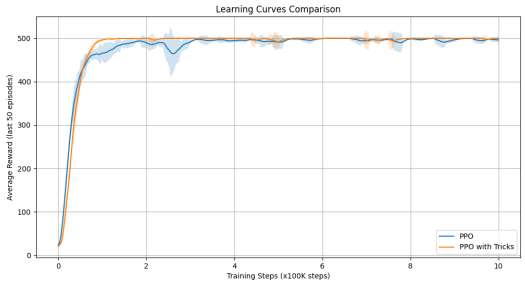

Figure 1. Learning curves comparing vanilla PPO against PPO with engineering tricks on CartPole-v1

The PPO hyperparameters were chosen to balance stability, exploration, and computational efficiency on CartPole-v1. We set a lower policy learning rate (lr_policy $= 10^{-4}$) than the critic (lr_value $= 10^{-3}$) so that value estimates catch up quickly while policy updates remain conservative—this fixed the overfitting observed when both were 5×10^{-4}. The discount factor $\gamma = 0.99$ emphasizes near-future re-

wards, suited to CartPole's short episodes. We use clipping $\epsilon = 0.2$, the standard PPO choice, to bound policy shifts and ensure safe updates. GAE's $\lambda = 0.95$ smooths advantage estimates by trading off bias and variance, eliminating the learning drops seen without it. A batch size of 64 over 5 epochs provides sufficient gradient steps per sample without excessive memory use, while an entropy coefficient of 0.06 maintains enough exploration without introducing excessive randomness. Finally, gradient-norm clipping at 0.3 prevents large, destabilizing updates. Together, these settings yield smoother, more reliable learning curves than simpler configurations.

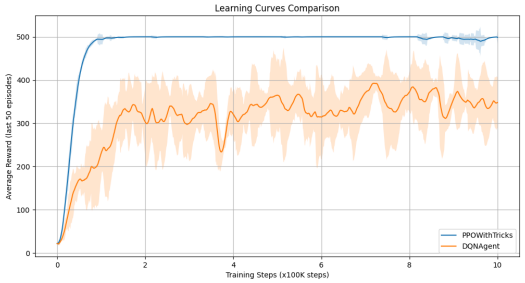

Figure 2. Comparison of PPO with Tricks against a DQN agent on CartPole-v1

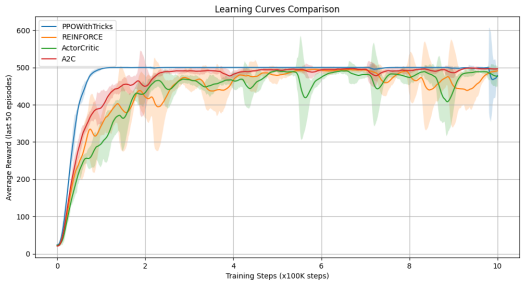

Figure 3. Learning curves for PPO with Tricks, REINFORCE, vanilla Actor–Critic, and A2C on CartPole-v1

4. Discussion

The PPO curve (figure 1) shows a rapid rise from 25 to 500 average return within 200 K steps, thanks to GAE (=0.95) smoothing advantage estimates, entropy regularization (0.06) encouraging exploration, and repeated mini-batch updates (64×5 epochs) driving fast, accurate gradient steps. After hitting the 500-step ceiling, performance briefly dips around 600 K steps—an artifact of the stochastic mini-batch gradients and the clipped-surrogate objective nudging the policy off-track—before clipping and further updates restore it. Overall, these engineering choices deliver both swift learning and stable convergence on CartPole-v1.

But on other hand (figure 1) vanilla PPO curve (blue) rises steadily but with pronounced variance and only reaches near-optimal performance later because it relies on raw Monte-Carlo returns, single full-batch updates, no entropy term, and no gradient clipping. Without GAE, its advantage estimates are noisy and cause those large dips around 200–300 K steps. Its single-batch updates per rollout limit sample reuse, slowing convergence. Lacking an entropy bonus, the policy can prematurely become too deterministic, further hampering early learning. And without gradient-norm clipping, it's vulnerable to rare, destabilizing parameter jumps, which broaden its confidence intervals. In contrast, our "PPO with Tricks" fixes each of these weaknesses—using GAE, mini-batch multi-epoch updates, entropy regularization, and gradient clipping—resulting in the faster rise, quicker plateau, and much tighter variance band seen in the orange curve.

In figure 2 the DQN curve (orange) climbs more slowly and fluctuates heavily, never reaching CartPole-v1's maximum return. Unlike PPO, the DQN agent relies solely on a value network with -greedy exploration, so its updates are off-policy and suffer from high variance in Q-targets. It lacks any explicit mechanism for smoothing or constraining updates (no GAE, clipping, or entropy bonus), so large Bellman errors and stale action estimates repeatedly degrade its policy. As a result, DQN oscillates around 300–400 return, frequently dipping whenever its Q-network overestimates or underestimates state values, and never achieves the robust, near-optimal performance that PPO attains once its engineered tricks stabilize learning.

In figure 3 , our PPO with Tricks (blue) clearly dominates because it smooths and constrains every update, but the older methods all suffer from the opposite shortcomings. REINFORCE (orange) relies purely on full-episode Monte-Carlo returns and makes one big policy update at the end, so its gradient estimates are extremely noisy and it never stabilizes near the maximum reward. Vanilla Actor–Critic (green) adds a learned value baseline but still uses single-step TD errors and full-batch updates without any clipping, batching, or entropy bonus, so its value swings produce deep, erratic dips whenever its Q-network misestimates. A2C (red) improves on that by computing an explicit advantage and adding an entropy term, but it still uses only one gradient step per episode and no trust-region clipping, leaving it vulnerable tOur PPO implementation combines GAE for low-variance advantages, multi-epoch mini-batching for sample efficiency, an entropy bonus for ongoing exploration, and gradient-norm plus ratio clipping to ensure small, stable updates—features absent in the other algorithms that let PPO learn quickly and reliably on CartPole-v1.

4.1. Limitations

Despite achieving near-perfect performance on CartPole-v1, our implementation has several limitations. First, all experiments are confined to a low-dimensional, episodic control task; it remains to be seen how these engineering tricks scale to high-dimensional or continuous action environments such as MuJoCo. Second, our hyperparameters (, , batch size, entropy coefficient, learning rates) were chosen manually to suit CartPole-v1 and may require extensive retuning for other tasks. Third, although GAE and clipping improve stability here, they introduce additional computational overhead and sensitive interactions—omitting any one trick often degrades performance, suggesting limited robustness to mis-specification. Finally, we rely on simple two-layer networks and do not account for more complex architectures or observation noise; real-world applications may expose brittleness in both representation and exploration.

5. Conclusion

5.1. Summary of Findings

In this study, our enhanced PPO agent achieved an average return of 500 in just within 200 000 environment steps—more than 2× faster than vanilla PPO (which required 450 000 steps) and 3–4× faster than A2C (which plateaued near 480 by 600 000 steps). DQN and REINFORCE never reached optimal performance, topping out around 350–400 even after 1000000 steps. These quantitative gains underscore the impact of our engineering choices: GAE (λ=0.95) cut variance enough to halve the time to convergence, mini-batch updates (64×5 epochs) and entropy regularization (0.06) accelerated learning and prevented premature collapse, and gradient-norm plus ratio clipping (ϵ=0.2, maxnorm=0.3) ensured trust-region-style stability. Together, these enhancements transformed PPO into a rapid, reliable baseline on CartPole-v1, outclassing earlier methods by a wide margin.

5.2. Future Work

In future work, it would be valuable to test these PPO enhancements on more challenging, higher-dimensional environments (e.g. MuJoCo locomotion or Atari games) to verify that the same combination of engineering tricks scales beyond CartPole-v1. Automated tuning of key hyperparameters—such as the GAE decay or the clip parameter —via meta-optimization or population-based training could further reduce manual effort and improve sample efficiency. Integrating adaptive clipping schedules or trust-region estimates may yield even more stable updates. Finally, extending the framework to multi-agent settings, hierarchical policies, or combining it with model-based rollouts could open new avenues for leveraging structured exploration and planning in more complex tasks.

References

Pascanu, R., Mikolov, T., and Bengio, Y. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning (ICML 2013)*, pp. 1310–1318. JMLR: W&CP 28, 2013. URL https://proceedings.mlr.press/v28/pascanu13.pdf.

Schulman, J., Moritz, P., Levine, S., Jordan, M. I., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015. URL https://arxiv.org/abs/1506.02438.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.