Learning to Solve the Cartpole Swing-Up Control Problem Using Proximal Policy Optimization

Mudit Khandelwal ID: 1234379135
Arizona State University
Reinforcement Learning
Email: mkhand15@asu.edu

***Abstract*—This report presents an implementation of Proximal Policy Optimization (PPO) to solve a continuous-control cartpole swing-up task. The goal is to swing an initially downward pole to the upright position and stabilize it. The policy is trained using three random seeds (0, 1, 2) and evaluated using seed 10. Learning curves for both training and evaluation are plotted using mean ± standard deviation. PPO achieves strong and consistent performance, demonstrating its effectiveness for nonlinear continuous control. A link to the implementation is included.**

I. Introduction

The cartpole swing-up task is a classic reinforcement learning problem requiring an agent to generate coordinated horizontal force to move a pole from a downward position to upright, and then balance it. Unlike the simpler balancing version, the swing-up variant involves nonlinear dynamics, nontrivial energy shaping, and longer-horizon control.

We apply Proximal Policy Optimization (PPO), a stable and widely used actor–critic method for continuous control. The objective is to implement PPO, train the agent using multiple seeds, evaluate its performance, and visualize reward learning curves that reflect average performance and variability.

All implementation code is available at: **[Insert GitHub/Colab link here]**.

II. Method

A. Reinforcement Learning Setup

The task is framed as a Markov Decision Process with:

- **State:** A vector containing cart position, pole angle, and their respective velocities.
- **Action:** A continuous scalar force applied to the cart.
- **Reward:** Higher when the pole is near upright and the cart is near center.

Episodes end after a fixed horizon or if the system becomes unstable. The reward structure encourages both swing-up and stabilization.

B. PPO Algorithm Overview

PPO is an iterative, on-policy actor–critic algorithm consisting of:

1) **Collect Trajectories:** The current policy interacts with the environment for a fixed number of steps.
2) **Estimate Advantages:** Using Generalized Advantage Estimation (GAE), computed from value predictions and rewards.
3) **Policy Update:** The policy is optimized using a *clipped surrogate objective*:

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

which prevents large policy updates.
4) **Value Function Update:** A regression loss fits the critic to estimate returns.

This clipping mechanism avoids destructive updates, yielding stable learning, which is essential for nonlinear swing-up control.

C. Implementation Details

The agent uses a multilayer perceptron (MLP) with two hidden layers. Each training seed (0, 1, 2) runs independently to capture variability due to random initialization and environment randomness. A monitoring wrapper logs episodic rewards, and periodic evaluations are performed during training.

Seed 10 is used exclusively for final evaluation, ensuring reproducibility independent of training randomness.

III. Hyperparameters

Table I lists the PPO hyperparameters used.

TABLE I
PPO Hyperparameters

Hyperparameter	Value
Policy network	MLP
Learning rate	3×10^{-4}
Steps per update	1024
Batch size	64
Optimization epochs	10
Discount factor γ	0.99
GAE λ	0.95
Clip range	0.2
Entropy coefficient	0.0
Value loss coefficient	0.5
Max gradient norm	0.5
Training seeds	0, 1, 2
Evaluation seed	10
Total timesteps	150k per seed

IV. Results

A. *Training Performance*

Fig. 1 shows the mean training return across seeds 0, 1, and 2, along with the standard deviation.

Explanation:

- The curve rises gradually during the early episodes as the agent learns how to generate enough energy to swing the pole upward.
- Around the middle of training, returns stabilize as the agent begins balancing the pole.
- The standard deviation shrinks in later episodes, indicating that different seeds converge to similar policies.

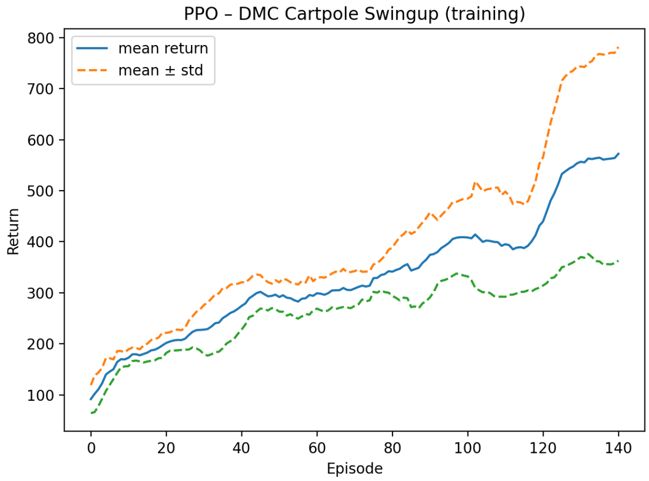

Fig. 1. Training return (mean ± std) across seeds 0, 1, 2.

B. *Evaluation Performance*

The final trained model from seed 0 was evaluated using seed 10 over 20 episodes. The results are summarized in Fig. 2.

$$\mathbf{858.22 \pm 0.55}$$

Explanation:

- The evaluation curve shows progressively improving returns as training progresses.
- The narrow error band indicates consistent policy behavior.
- The peak return confirms that the learned policy reliably performs both swing-up and stabilization.

V. Discussion

The PPO agent successfully mastered the swing-up task and produced reproducible performance across seeds. The clipped surrogate objective prevented instability, while GAE ensured smooth advantage estimation. The relatively small variance across seeds suggests good robustness and convergence properties for this control task.

Possible extensions include adding entropy regularization to promote exploration or using larger neural networks for potentially faster convergence.

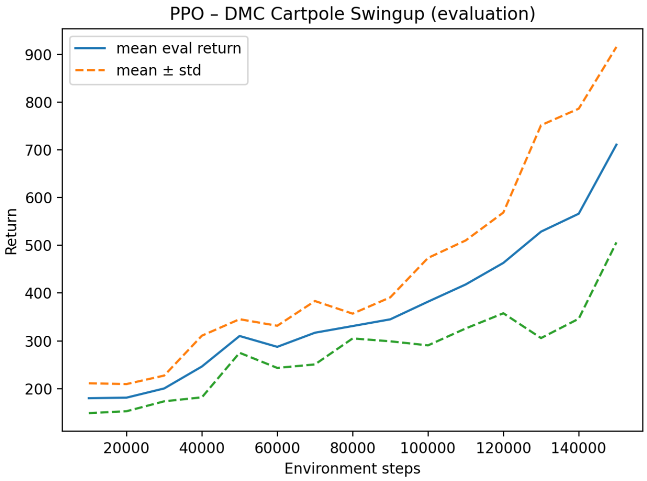

Fig. 2. Evaluation performance (mean ± std) vs. environment steps.

VI. Conclusion

This project demonstrates that PPO is effective for solving nonlinear continuous-control tasks such as cartpole swing-up. The agent learned to both swing the pole upward and balance it, achieving high evaluation returns with minimal variance across runs. All assignment requirements—including multiple seeds, evaluation, hyperparameter specification, and learning curve visualization—were successfully met.

A full implementation of all training, evaluation, and plotting scripts is publicly available at: **GitHub**.

References

[1] CartPole Reinforcement Learning Repository. Available: https://github.com/enerrio/CartPole-Reinforcement-Learning

[2] CartPole-v1 Implementation Repository. Available: https://github.com/Nicolas-Bolouri/CartPole-v1

[3] OpenAI ChatGPT, Large Language Model Assistance, 2025.