# PROXIMAL POLICY OPTIMIZATION ALGORITHMS

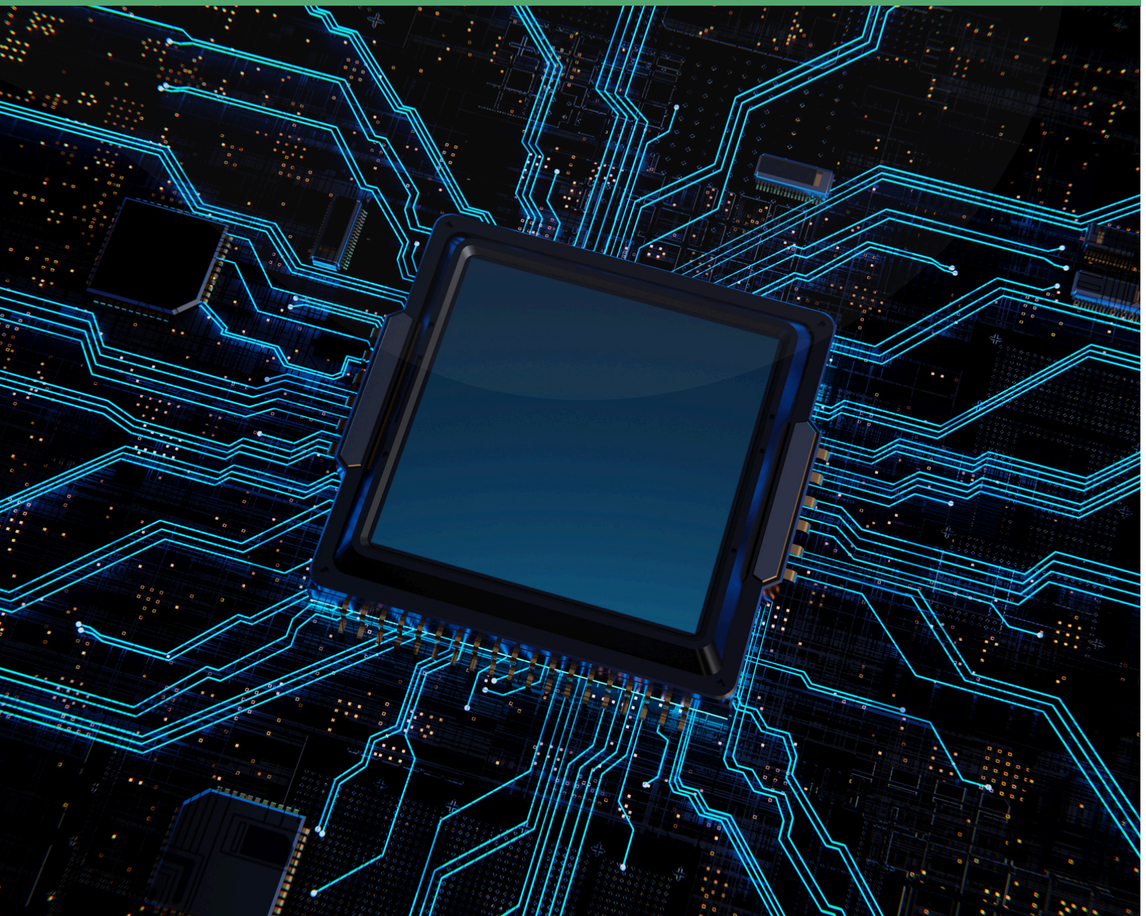
Nguyen Huu Hoang Long

Le Trong Dai Truong

# TABLE OF CONTENT


# PART I: INTRODUCTION

# Introduction



## The Impact of Reinforcement Learning

Reinforcement Learning is rapidly transforming various aspects of our lives, making it a true revolutionary force in AI.

## The Power of Trial and Error

Unlike traditional methods where machines are spoon-fed data, Reinforcement Learning (RL) mimics how humans learn: through trial and error, with rewards for good decisions and nudges in the right direction for bad ones.

# APPROACHING

In reinforcement learning (RL), algorithms can generally be categorized into two main types: policy-based and value-based methods. Each approach has its own strengths and weaknesses and is suitable for different kinds of problems.

## 1 POLICY-BASED

The goal is to directly learn the optimal policy, denoted as π*

- **REINFORCE.**

## 2 VALUE-BASED

The focus is on learning the optimal value function, denoted Q* or V*

- **Q-learning.**
- **SARSA.**

# PROBLEM

**Gradient-based policy** methods such as REINFORCE often encounter difficulties in updating policies that are unstable, leading to sudden policy changes and causing "catastrophic forgetting" or rapidly decreasing performance.

**Vanishing/Exploding Gradients:** Gradients can become too large or too small, leading to instability in the learning process.

# INTUITION

**Improving the training stability of the policy by limiting the change you make to the policy at each training epoch: avoiding having too large of a policy update.**

**Motivation:**

Smaller policy updates during training are **more likely to converge to an optimal solution.**

A too-big step can result in "falling off the cliff" **and taking a long time or even having no possibility to recover.**

## Proximal Policy Optimization


# PART II: METHODOLOGY

# POLICY GRADIENT METHODS

Policy gradient methods work by computing an estimator of the policy gradient and plugging it into a stochastic gradient ascent algorithm.

## Policy Objective Function

$$L^{PG}(\theta) = \hat{E}_t[log\pi_\theta(a_t|s_t)\hat{A}_t]$$

log probability of taking that action at that state

advantage if A > 0, this action is better than other action posible at that state

The idea was that by taking a gradient ascent step on this function, **we would push our agent to take actions that lead to higher rewards and avoid harmful actions**.

# POLICY GRADIENT METHODS

**Policy Objective Function**

$$L^{PG}(\theta) = \hat{E}_t[log\pi_\theta(a_t|s_t)\hat{A}_t]$$

log probability of taking that action at that state

advantage if A > 0, this action is better than other action posible at that state

## However, the problem comes from the step size:

- Too small, the training process was too slow.
- Too high, there was too much variability in the training.

## Policy Update Constraints

Using an objective function designed to limit the extent of the policy update, effectively keeping the policy updates within a "safe" range.

# TRUST REGION POLICY OPTIMIZATION (TRPO)

### Constrained Optimization

TRPO modifies the standard policy gradient update by incorporating a constraint on the **Kullback-Leibler (KL)** divergence that ensures the new policy does not deviate too much from the old policy.

### Trust Region

The idea of a trust region is to restrict the updates to the policy to a "safe" region where the new policy is not too far from the old policy.

### Objective Function

The objective function in TRPO is designed to maximize the expected return while keeping the KL divergence between the new policy and the old policy below a specified threshold.

### Advantage

This helps in maintaining stable learning and avoiding drastic policy changes that can lead to poor performance.

# Trust Region Policy Optimization

$$maximize_\theta = \hat{E}_t[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t]$$

**Subject to:**

$$\hat{E}_t[KL[\pi_{\theta_{old}}(.|s_t), \pi_\theta(.|s_t)]] \leq \delta$$

**Where:**

$\theta_{old}$ : The vector of policy parameters before update.

$KL$ : The Kullback-Leibler divergence.

$\delta$ : Small positive number representing the trust region constraint.

# THE KULLBACK–LEIBLER DIVERGENCE

## Entropy

Given a discrete random variable $X$ taking values $\{x_1, \ldots, x_n\}$ and the probability mass function (pmf) $\Pr(X)$, the Entropy of $X$ is:

$$H(X) = \mathbb{E}(I(X)) = \sum_{i=1}^{n} \Pr(x_i) I(x_i) = \sum_{i=1}^{n} \Pr(x_i) \log \frac{1}{\Pr(x_i)}$$

In other words:

$$H(X) = -\sum_{i=1}^{n} \Pr(x_i) \log \Pr(x_i)$$

Sometimes, for convenience and better readability, we can write entropy with the probability vector $\mathbf{p} = (p_1, \ldots, p_n)$ where $p_i = \Pr(X = x_i)$ then $H(\mathbf{p})$:

$$H(\mathbf{p}) = -\sum_{i=1}^{n} p_i \log p_i$$

# THE KULLBACK–LEIBLER DIVERGENCE

## Cross Entropy

Given two discrete probability distributions $P$ and $Q$ and the corresponding probability vectors of the distributions $\mathbf{p} = (p_1, \ldots, p_n)$ and $\mathbf{q} = (q_1, \ldots, q_n)$, the Cross Entropy is defined as follows:

$$H(\mathbf{p}, \mathbf{q}) = -\sum_{i=1}^{n} p_i \log q_i$$

# THE KULLBACK–LEIBLER DIVERGENCE

## Kullback–Leibler divergence

Given two discrete probability distributions $P$ and $Q$ and the corresponding probability vectors of the distributions $\mathbf{p} = (p_1, \ldots, p_n)$ and $\mathbf{q} = (q_1, \ldots, q_n)$, for a discrete random variable $X$ taking values $\{x_1, \ldots, x_n\}$, the Kullback–Leibler divergence is calculated as:

$$D_{KL}(p \parallel q) = H(p, q) - H(p)$$

In other words:

$$D_{KL}(p \parallel q) = -\sum_{i=1}^{n} p_i \log \frac{q_i}{p_i} = \sum_{i=1}^{n} p_i \log \frac{p_i}{q_i}$$

# PROXIMAL POLICY OPTIMIZATION (PPO)

### Unconstrained Optimization

TRPO involves solving a complex constrained optimization problem, making it computationally expensive and difficult to implement. PPO simplifies the optimization process while retaining TRPO's stability benefits.

### Objective Function

The objective function of PPO is designed to constrain policy updates. The key component of the PPO objective is the clipped surrogate objective function. It can be expressed as:

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

# PROXIMAL POLICY OPTIMIZATION

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

## The Ratio Function

The probability ratio between the new policy and the old policy.

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

As we can see **Ratio Function** denotes the probability ratio between the current and old policy:

- **If Ratio Function > 1:** The ***action_t*** at ***state_t*** is more likely in the current policy than the old policy.
- **If Ratio Function is between 0 and 1:** The ***action_t*** is less likely for the current policy than for the old one.

**So this probability ratio is an easy way to estimate the divergence between old and current policy.**

# PROXIMAL POLICY OPTIMIZATION

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

**The Unclipped Part**

This gives us the left part of the new objective function: multiplying the ratio by the advantage.

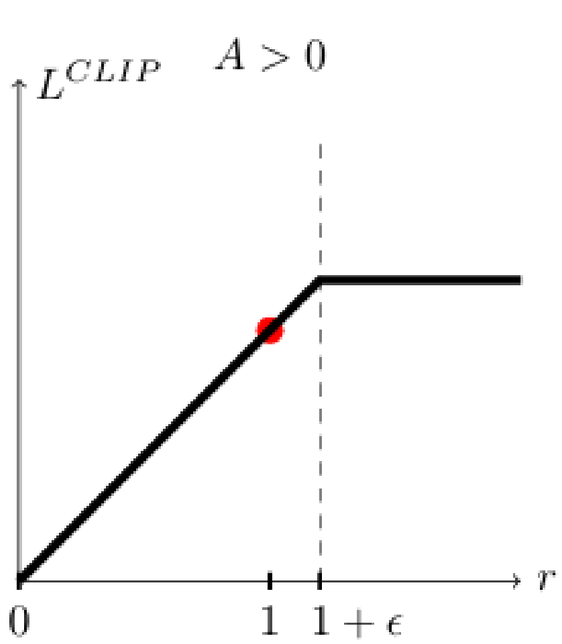
**The Clipped Part**

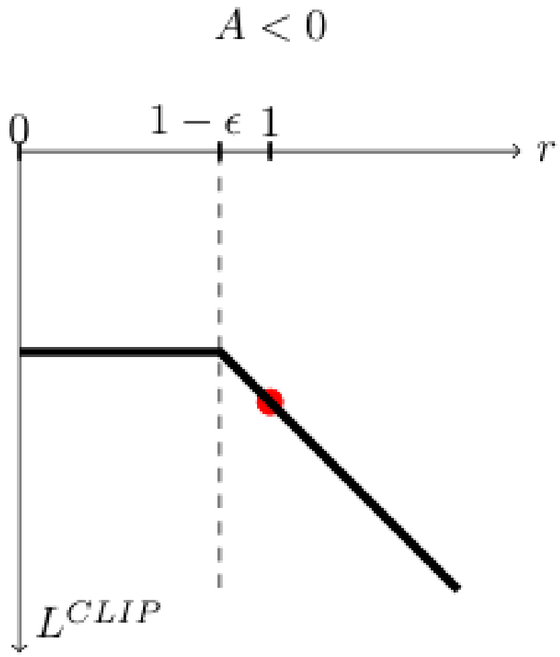
This clipped part is a version where $rt(\theta)$ is clipped between [1-ϵ,1+ϵ]. Epsilon is a hyperparameter that helps us to define this clip range.

**Then, we take the minimum of the clipped and non-clipped objective, so the final objective is a lower bound (pessimistic bound) of the unclipped objective.**

# PROXIMAL POLICY OPTIMIZATION

**Selecting either the clipped or the non-clipped objective based on the ratio and advantage situation.**



# PROXIMAL POLICY OPTIMIZATION

**The final Clipped Surrogate Objective Loss for PPO Actor-Critic style looks like this, it's a combination of Clipped Surrogate Objective function, Value Loss Function and Entropy bonus:**

$$L^{CLIP+VF+S}(\theta) = \hat{E}_t[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)]$$

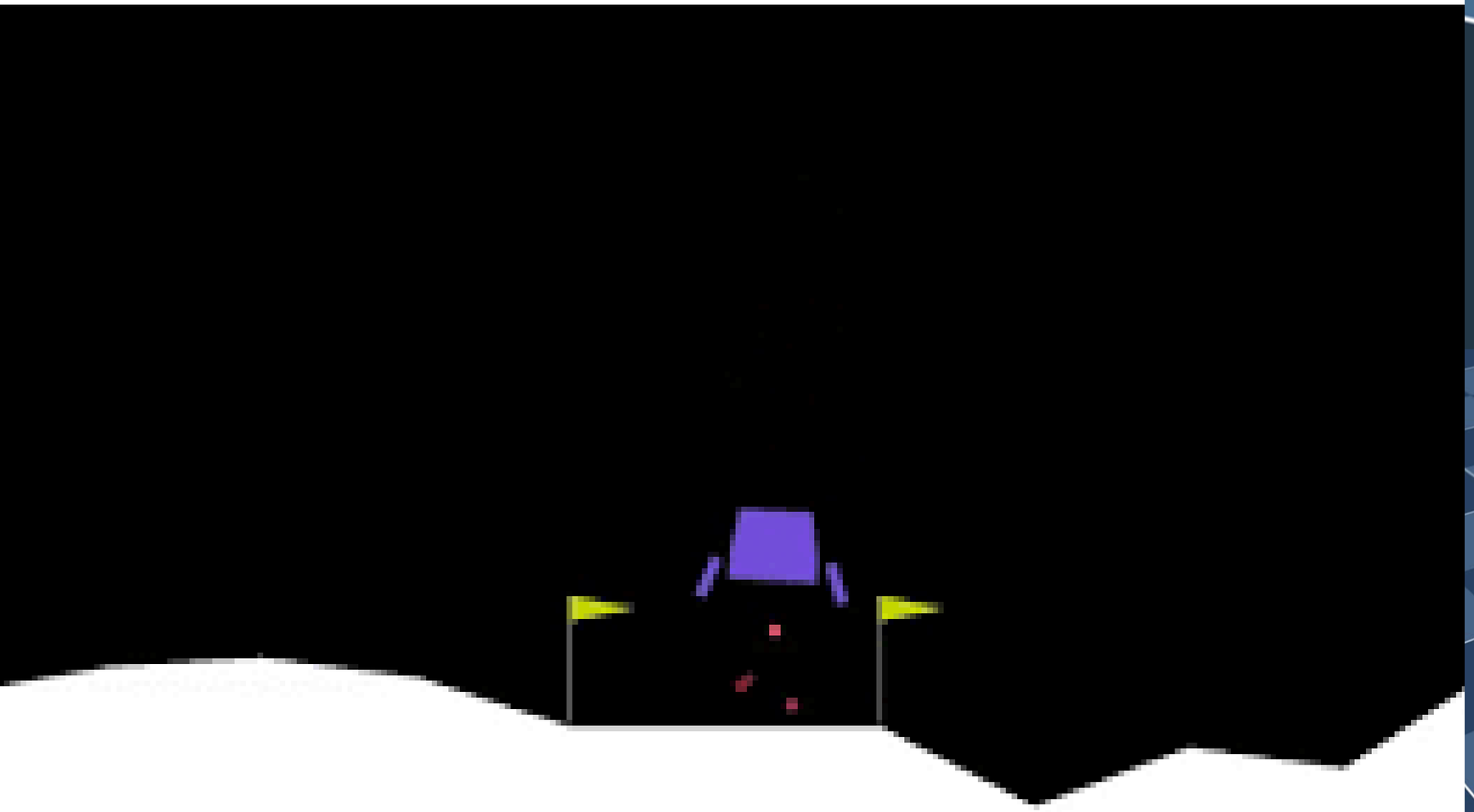
**Square-error value loss**

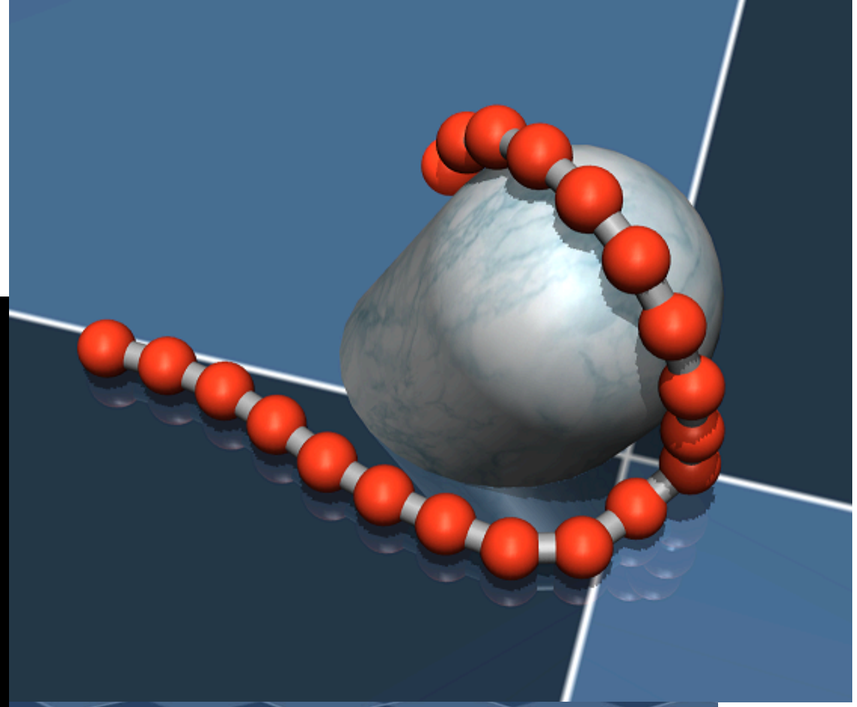
$$(V_\theta - V^{target})^2$$

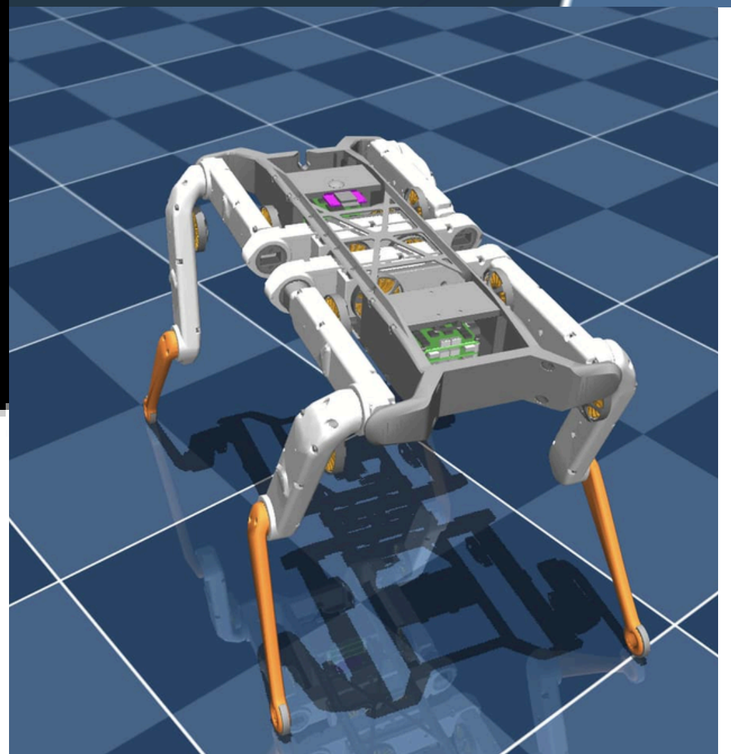
**Add an entropy bonus to ensure sufficient exploration**

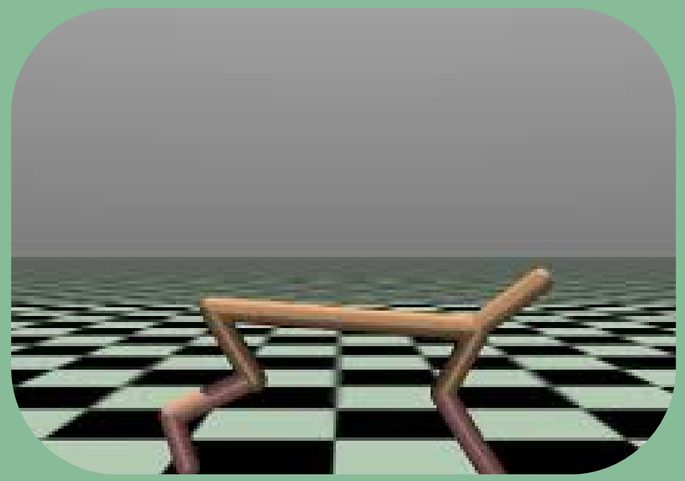
# PART III: EXPERIMENT

# Mujoco - Gymnasium

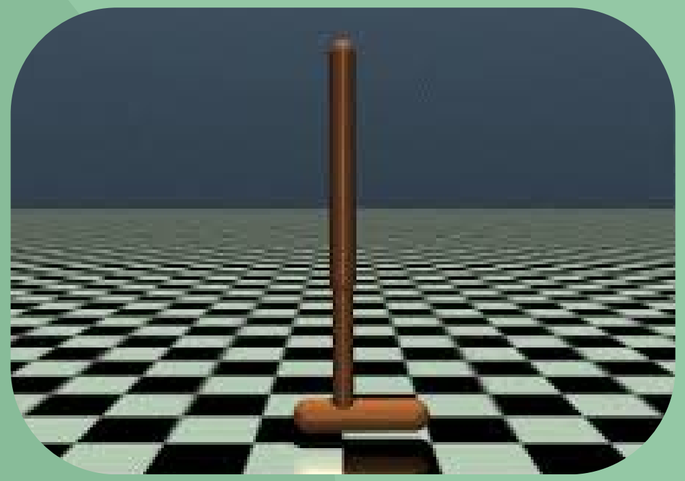




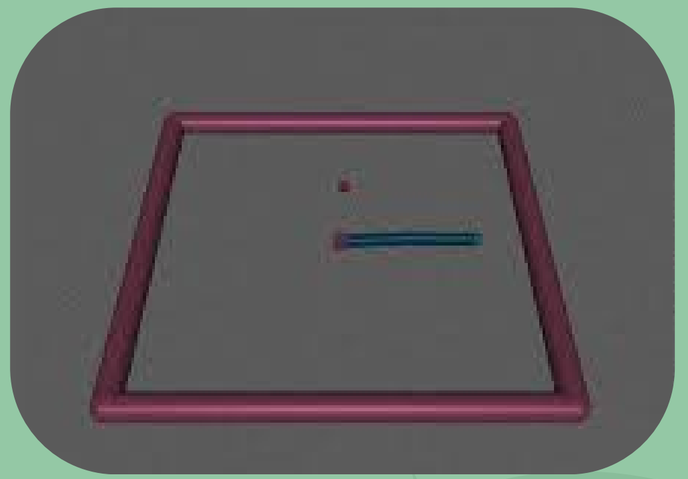




HALFCHEETAH



HOPPER



REACHER

HUMANOID STANDUP

HUMANOID

# STABLE BASELINES 3

- **Stable Baselines3 (SB3)** is a set of reliable implementations of reinforcement learning algorithms in PyTorch. It is the next major version of Stable Baselines.
- **SB3-Contrib:** providing the latest features, like TRPO.

# PROXIMAL POLICY OPTIMIZATION (PPO)

**Hyperparameters:**

- policy: MlpPolicy
- learning_rate: 0.0003
- n_steps: 2048
- batch_size: 64
- n_epochs: 10
- gamma: 0.99
- total_timesteps: 1000000

```
from stable_baselines3 import PPO


model = PPO("MlpPolicy", vec_env, verbose=1)
model.learn(total_timesteps=1000000)
model.save("")
```

# ADVANTAGE ACTOR CRITIC (A2C)

**Hyperparameters:**

- policy: MlpPolicy
- learning_rate: 0.0007
- n_steps: 5
- gamma: 0.99
- total_timesteps: 1000000

```
from stable_baselines3 import PPO


model = PPO("MlpPolicy", vec_env, verbose=1)
model.learn(total_timesteps=1000000)
model.save("")
```

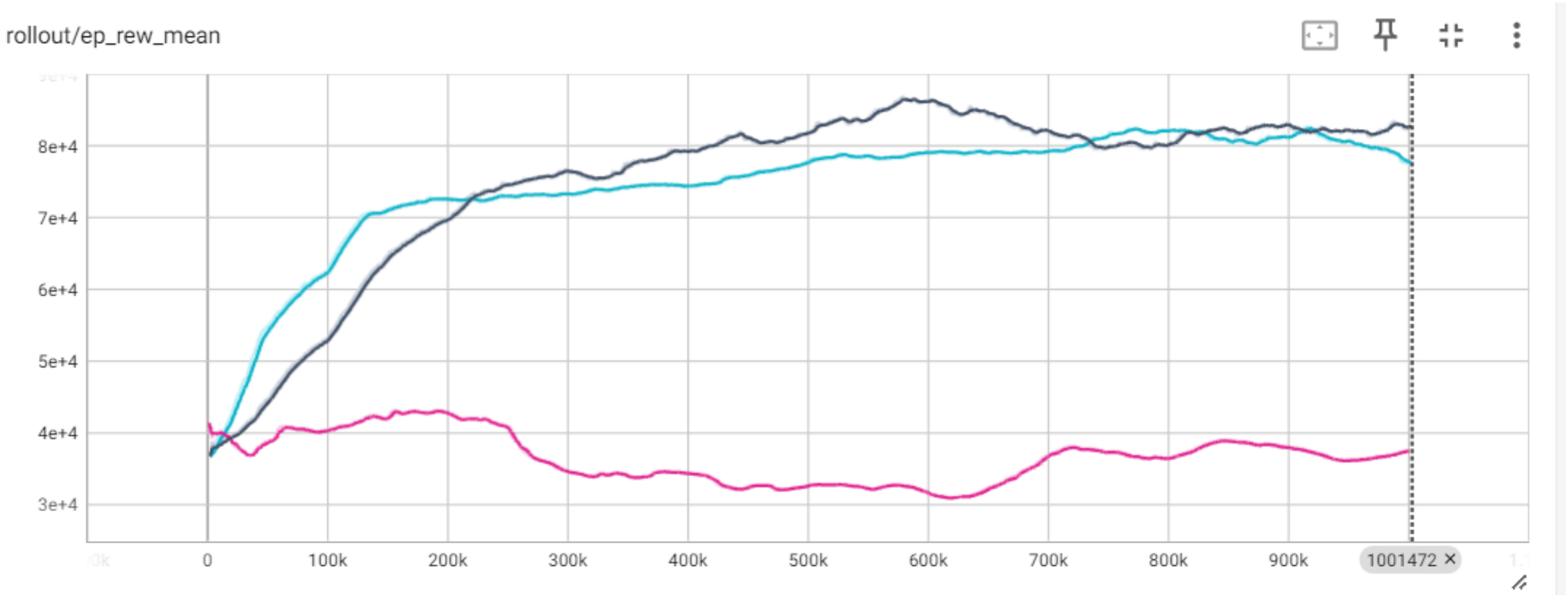
# TRUST REGION POLICY OPTIMIZATION (TRPO)

**Hyperparameters:**

- policy: MlpPolicy
- learning_rate: 0.001
- n_steps: 2048
- batch_size: 128
- gamma: 0.99
- total_timesteps: 1000000

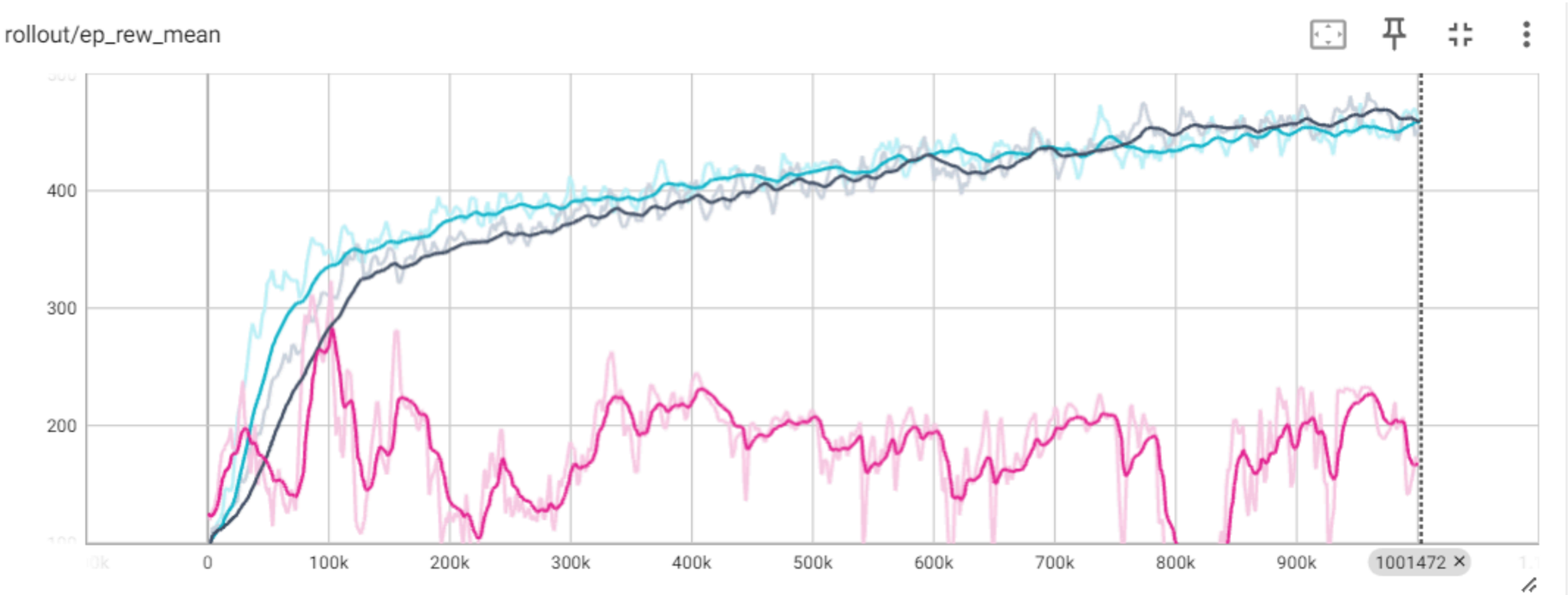
```
from sb3_contrib import TRPO


model = TRPO("MlpPolicy", env, verbose=1)
model.learn(total_timesteps=1000000)
model.save("")
```

# HUMANOID STANDUP



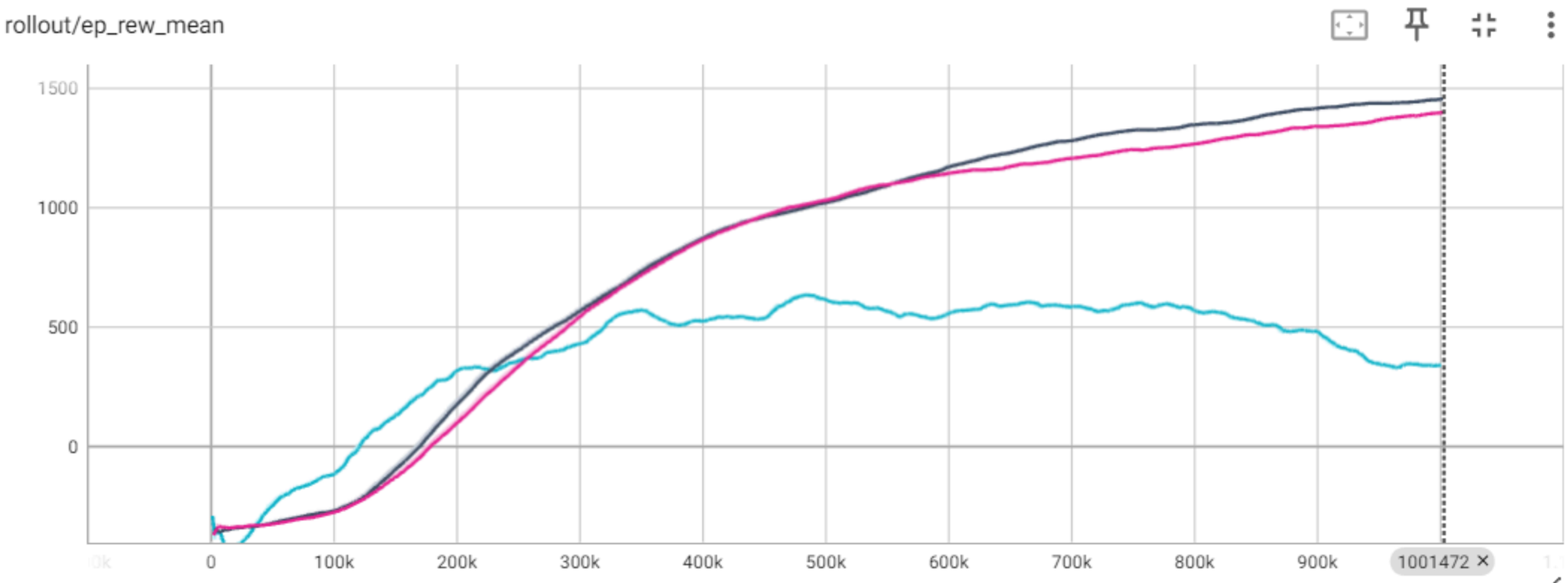

| Run | Smoothed | Value | Step | Relative |
|---|---|---|---|---|
| A2C_HumanoidStandupv4 | 37.585,2733 | 37.656,3945 | 1.000.000 | 42.74 min |
| PPO_HumanoidStandupv4 | 82.731,284 | 82.971,8594 | 1.001.472 | 45.99 min |
| TRPO_HumanoidStandupv4 | 77.630,2033 | 77.284,8594 | 1.001.472 | 34.85 min |

# HUMANOID



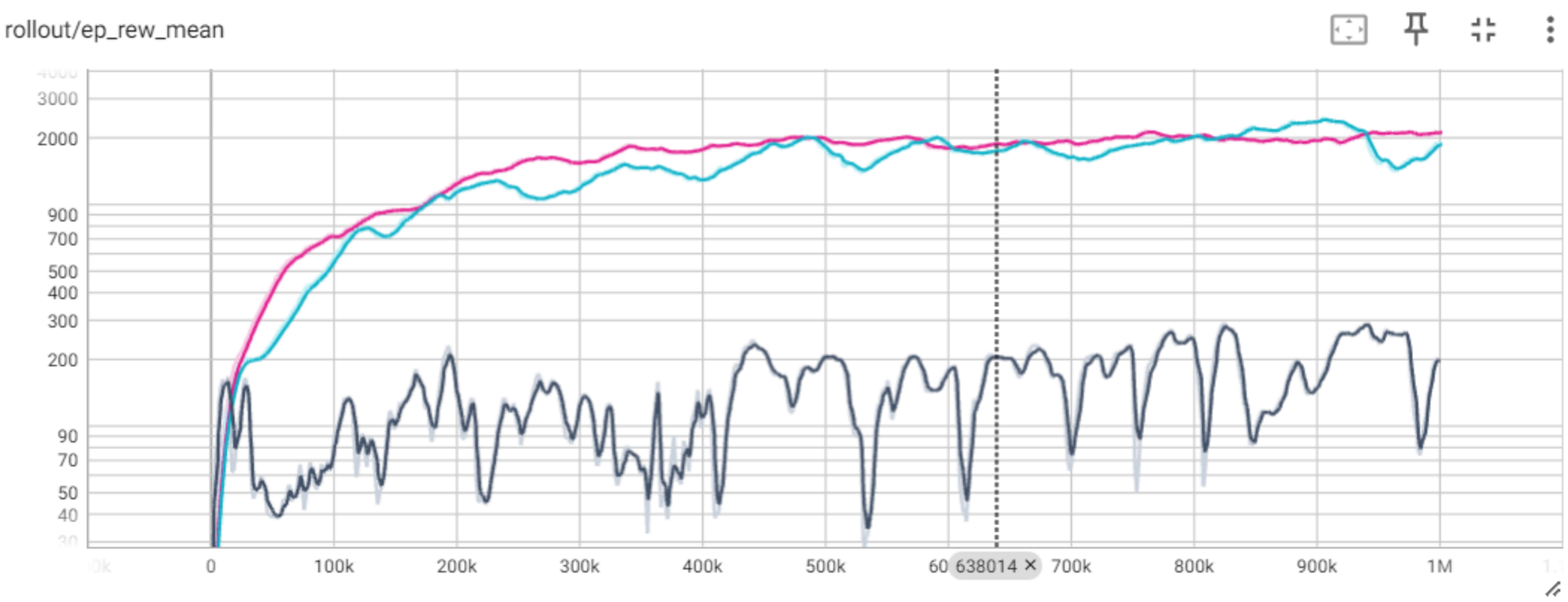

| Run ↑ | Smoothed | Value | Step | Relative |
|---|---|---|---|---|
| A2C_Humanoidv4 | 167,6359 | 168,0874 | 1.000.000 | 45.83 min |
| PPO_Humanoidv4 | 458,874 | 454,6227 | 1.001.472 | 45.41 min |
| TRPO_Humanoidv4 | 458,5069 | 457,4466 | 1.001.472 | 35.24 min |

# HALF CHEETAH



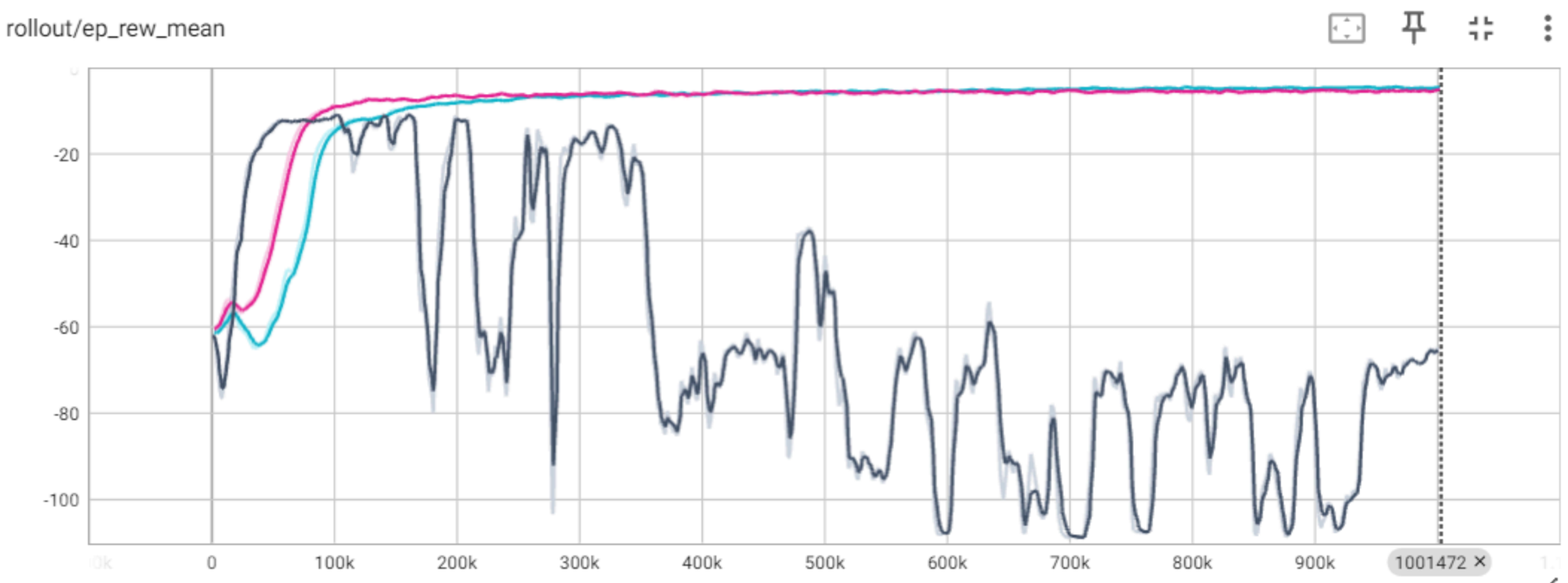

| Run ↑ | Smoothed | Value | Step | Relative |
|---|---|---|---|---|
| A2C_HalfCheetahv4 | 343,4118 | 347,5318 | 1.000.000 | 40.52 min |
| PPO_HalfCheetahv4 | 1.456,3258 | 1.460,3878 | 1.001.472 | 36.33 min |
| TRPO_HalfCheetahv4 | 1.399,7272 | 1.403,1122 | 1.001.472 | 24.1 min |

# HOPPER



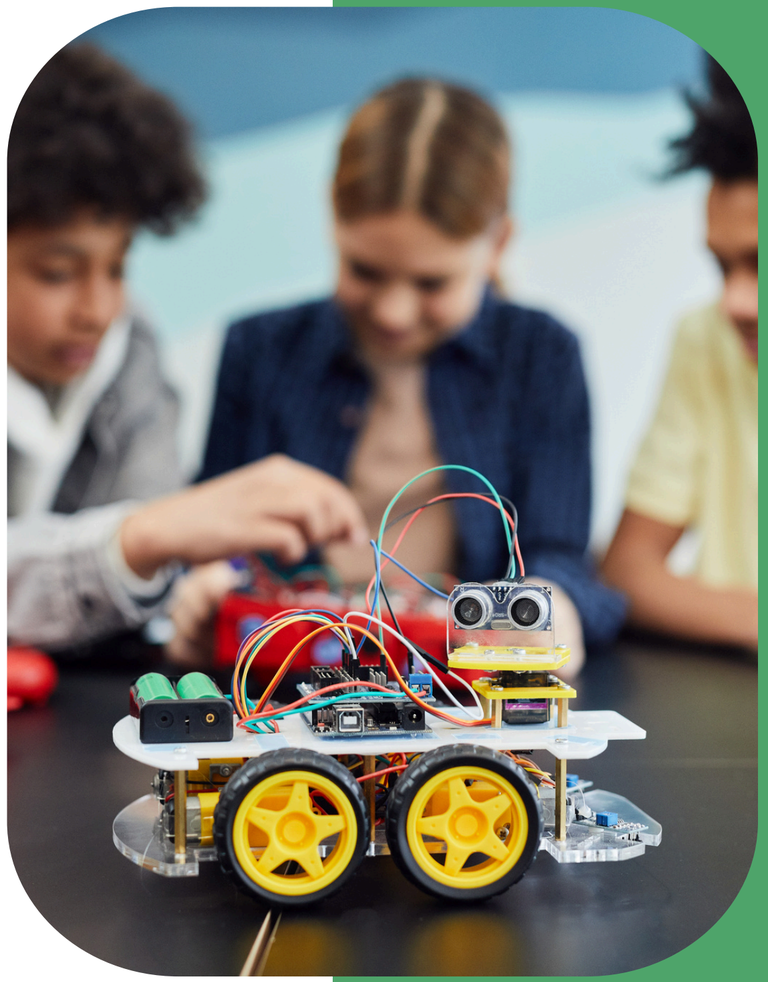

| Run ↑ | Smoothed | Value | Step | Relative |
|---|---|---|---|---|
| A2C_Hopperv4 | 205,8134 | 206,7789 | 638.500 | 27.1 min |
| PPO_Hopperv4 | 1.750,7809 | 1.778,8584 | 638.976 | 24.41 min |
| TRPO_Hopperv4 | 1.884,367 | 1.910,6417 | 638.976 | 17.17 min |

# REACHER




| Run ↑ | Smoothed | Value | Step | Relative |
|---|---|---|---|---|
| A2C_Reacherv4 | -65,3122 | -64,6839 | 1.000.000 | 28.76 min |
| PPO_Reacherv4 | -4,6227 | -4,6145 | 1.001.472 | 34.15 min |
| TRPO_Reacherv4 | -5,1061 | -5,006 | 1.001.472 | 23.73 min |

# PART IV: CONCLUSION



# Advantages

### Stability and Sample Efficiency

The clipping mechanism prevents drastic policy updates, ensuring smoother learning and convergence

### Simplicity and Ease of Implementation

PPO is known for being relatively easy to understand and implement compared to some other reinforcement learning algorithms.

### Strong Performance Across Tasks

PPO has a proven track record of achieving good performance on a wide variety of reinforcement learning tasks

# Limitations

**Sensitivity to Hyperparameters**

PPO relies on several hyperparameters, tuning these parameters significantly impacts the algorithm's performance.

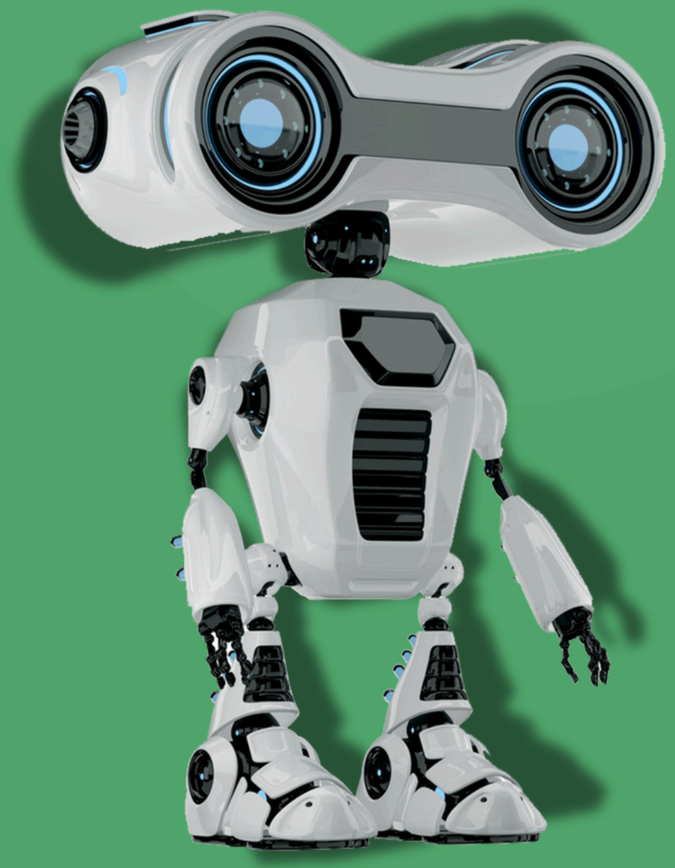
**Local Optima**

There's a possibility of PPO getting stuck in local optima.

**Exploration vs. Exploitation Trade-off**

PPO balances exploration (trying new actions) and exploitation (focusing on successful actions). However, this balance might not be perfect for all tasks.

# References




- https://arxiv.org/pdf/1707.06347v2
- https://huggingface.co/learn/deep-rl-course/unit8/introduction
- https://paperswithcode.com/method/ppo
- https://spinningup.openai.com/en/latest/algorithms/ppo.html
- https://gymnasium.farama.org/environments/mujoco/
- https://github.com/DLR-RM/stable-baselines3
- https://www.kaggle.com/code/mmdalix/openai-gym-mujoco-env-setup-and-training-2022#Step-2-:-Install-Virtual-Display

# THANK YOU

For Your Attention