ROBUST ADVERSARIAL REINFORCEMENT LEARNING

Andrea Baldi 1966232
Serena Trovalusci 2128733

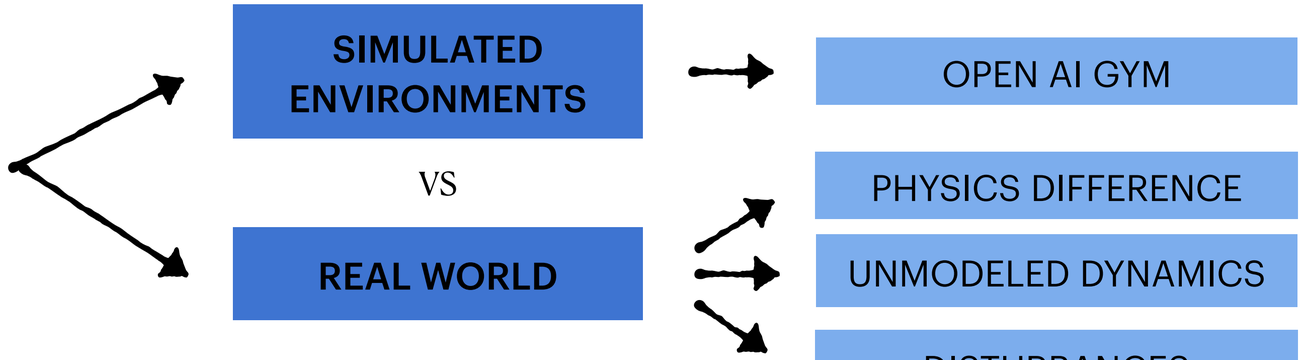
Introduction

- Deep Neural networks coupled with fast simulation and improved computation have led to recent successes in the field of **Reinforcement Learning:**

The Generalization Problem

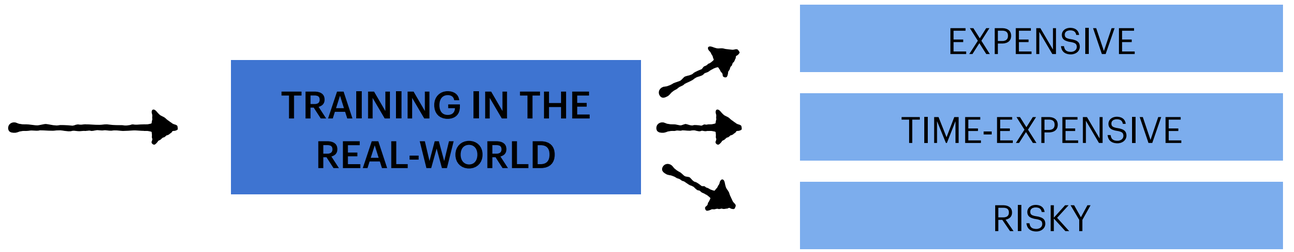
- Most **RL**-based approaches **fail to generalize** because of:

1. **SIMULATION-TO-REAL-GAP**



Policies trained in simulation often fail in the real world.

2. **REAL-WORLD DATA SCARSITY**



- RL training is often limited to a small number of training conditions (e.g., fixed friction, mass).
- When tested in new conditions the learned policy fails.

ROBUST ADVERSARIAL REINFORCEMENT LEARNING (RARL)

- We train a protagonist agent (main agent) to operate in the presence of an adversary agent that applies disturbance forces to the system.
- The adversary is also an agent that learns to maximize the optimal destabilization policy.
- The protagonist must learn to be robust against the adversary.
- "Zero-sum Markov game": the protagonist tries to maximize its reward function while the adversary tries to minimize it by applying optimal disturbances.

Two-player zero-sum discounted games

- We express the adversarial setting as a two player γ discounted zero-sum Markov game: $(S, A_1, A_2, P, r, \gamma, s_0)$. Player 1 is playing with strategy (policy) μ while player 2 is playing with strategy (policy) v.
- A_1, A_2 continuous set of actions the players can take.
- $P : SxA_1xA_2xS \rightarrow R$ transition probability density function.
- $r : SxA_1xA_2 \rightarrow R$ reward of both players: $r_{\mu,v} = E_{a_1 \sim \mu(.|s), a_2 \sim v(.|s)}[r(s, a_1, a_2)]$
- Player 1 maximises the γ discounted reward while player 2 minimizes it.

Formulating Adversarial Reinforcement Learning

- In the adversarial game, at every timestep t both players observe the state s_t and take actions $a_{1,t} \sim \mu(s_t)$ and $a_{2,t} = \nu(s_t)$.
- The state transitions $s_{t+1} = P(s_t, a_{1,t}, a_{2,t})$ and the reward $r_t = r(s_t, a_{1,t}, a_{2,t})$ is obtained from the environment.
- The protagonist gets a reward $r_{1,t} = r_t$ while the adversary gets a reward $r_{2,t} = -r_t$
- Each step is represented as: $(s_t, a_{1,t}, a_{2,t}, r_{1,t}, r_{2,t}, s_{t+1})$
- The protagonist tries to maximize: $R_1 = E_{a_1 \sim \mu(s), a_2 \sim \nu(s)}[\sum_t r_1(s, a_1, a_2)]$
- The adversary attempts to maximize: $R_2 = -R_1$

Proposed Method:
RARL algorithm

Input: Environment $\mathcal{E}$; Stochastic policies μ and ν
Initialize: Learnable parameters θ_0^μ for μ and θ_0^ν for ν
for i=1,2,..N_{iter} **do**
  $\theta_i^\mu \leftarrow \theta_{i-1}^\mu$
  for j=1,2,..N_μ **do**
    $\{(s_t^i, a_t^{1i}, a_t^{2i}, r_t^{1i}, r_t^{2i})\} \leftarrow \text{roll}(\mathcal{E}, \mu_{\theta_i^\mu}, \nu_{\theta_{i-1}^\nu}, N_{\text{traj}})$
    $\theta_i^\mu \leftarrow \text{policyOptimizer}(\{(s_t^i, a_t^{1i}, r_t^{1i})\}, \mu, \theta_i^\mu)$
  end for
  $\theta_i^\nu \leftarrow \theta_{i-1}^\nu$
  for j=1,2,..N_ν **do**
    $\{(s_t^i, a_t^{1i}, a_t^{2i}, r_t^{1i}, r_t^{2i})\} \leftarrow \text{roll}(\mathcal{E}, \mu_{\theta_i^\mu}, \nu_{\theta_i^\nu}, N_{\text{traj}})$
    $\theta_i^\nu \leftarrow \text{policyOptimizer}(\{(s_t^i, a_t^{2i}, r_t^{2i})\}, \nu, \theta_i^\nu)$
  end for
end for
Return: $\theta_{N_{\text{iter}}}^\mu, \theta_{N_{\text{iter}}}^\nu$

Environment Extension

ENVIRONMENT

↓

INVERTED PENDULUM-V5 / WALKER2D-V5 (Gymnasium)

+

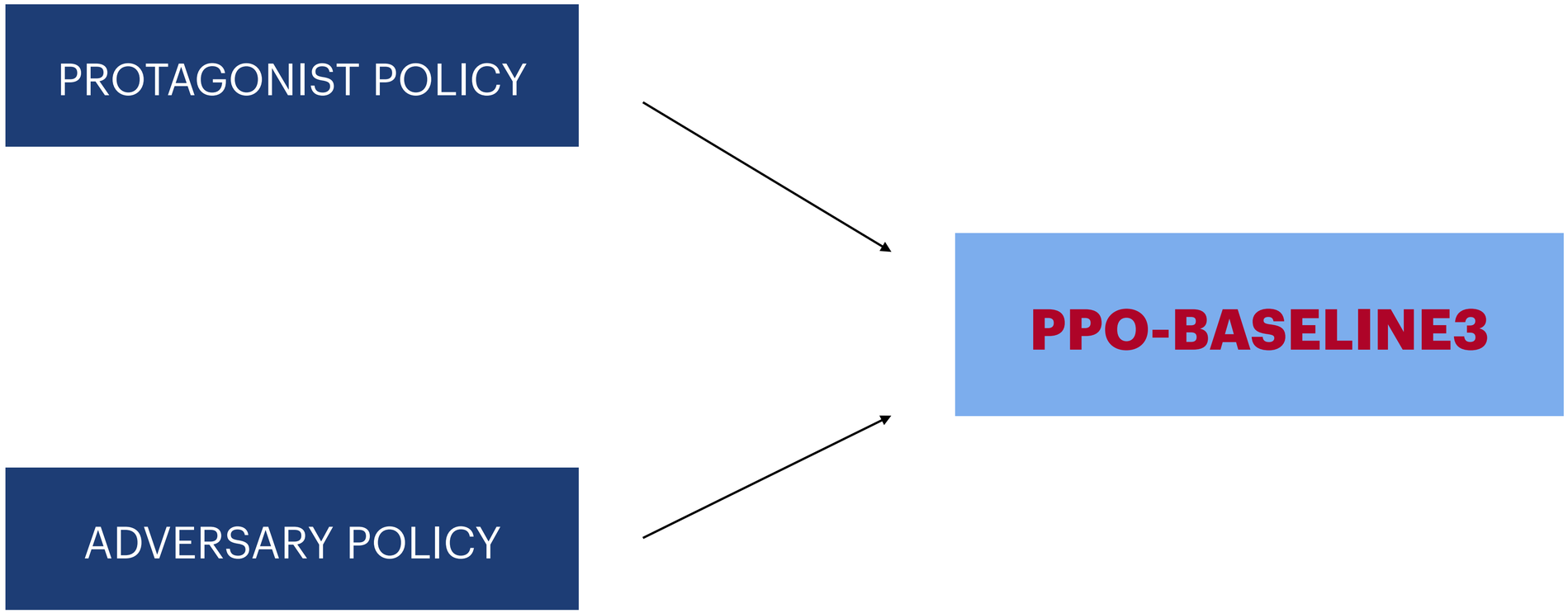
1. **Policies included** in the environment (protagonist, adversary)
2. **Locking function** that decides the current training policy

Environment Extension

STEP FUNCTION

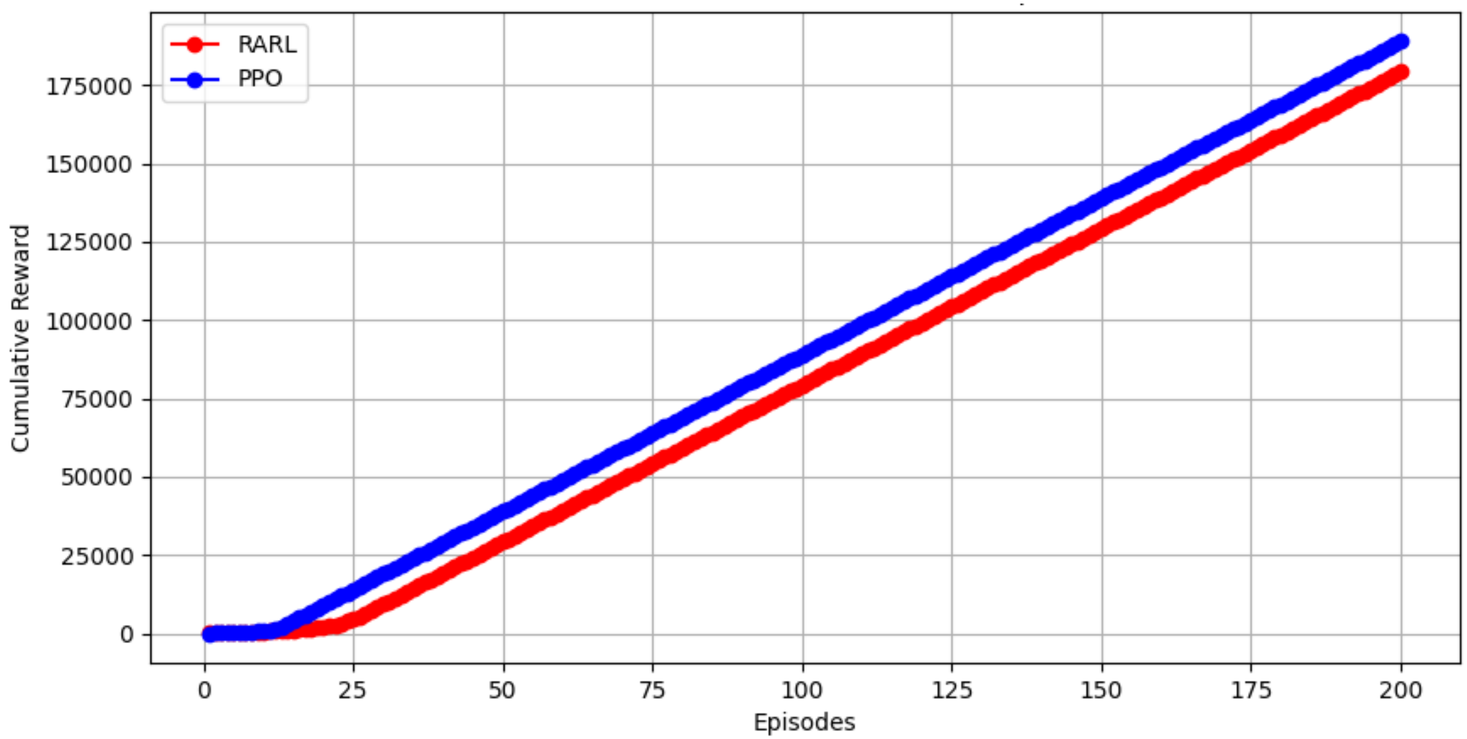
1. Provides the action for the “resting” policy.
2. Sums adversary and protagonist actions.
3. Clips the sum of the actions.
4. For Walker2d: Redefines reward.
5. Assigns reward to the protagonist and the adversary (R_a=-R_p)

Policy Optimization



Inverted Pendulum

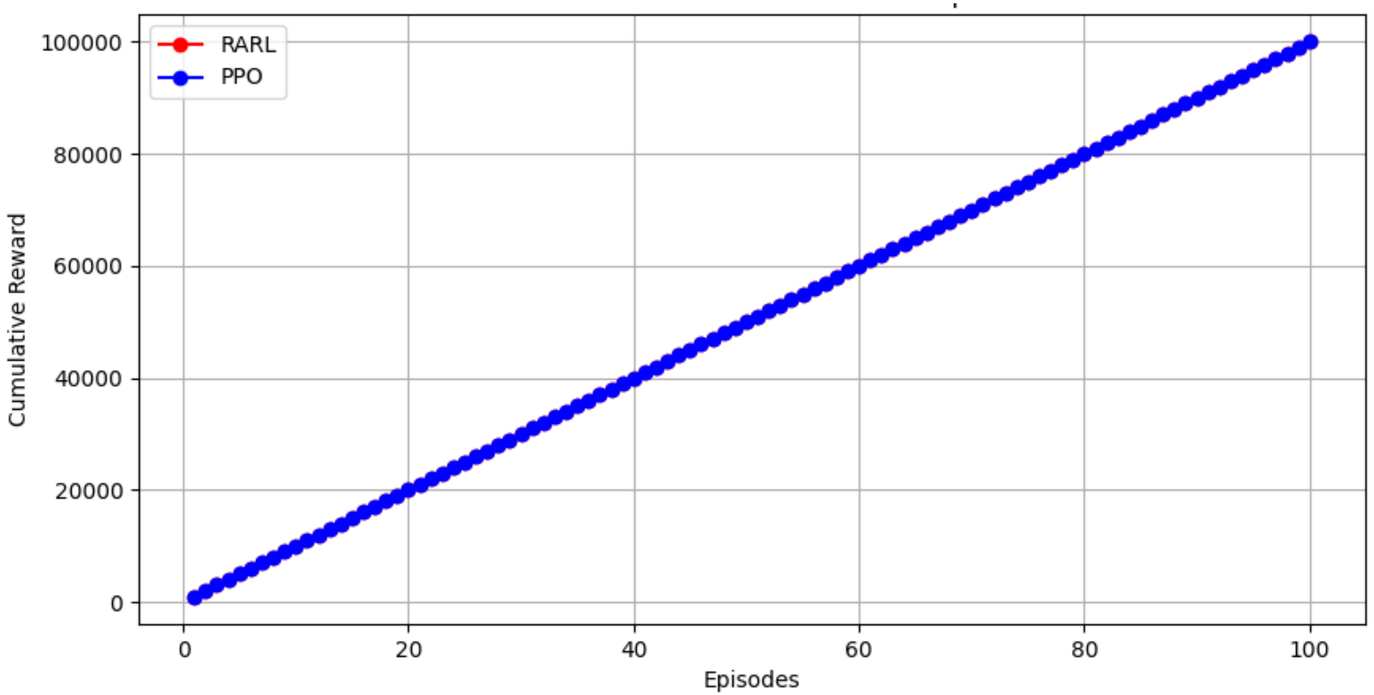
“InvertedPendulum-v5”



CUMULATIVE REWARD DURING TRAINNING

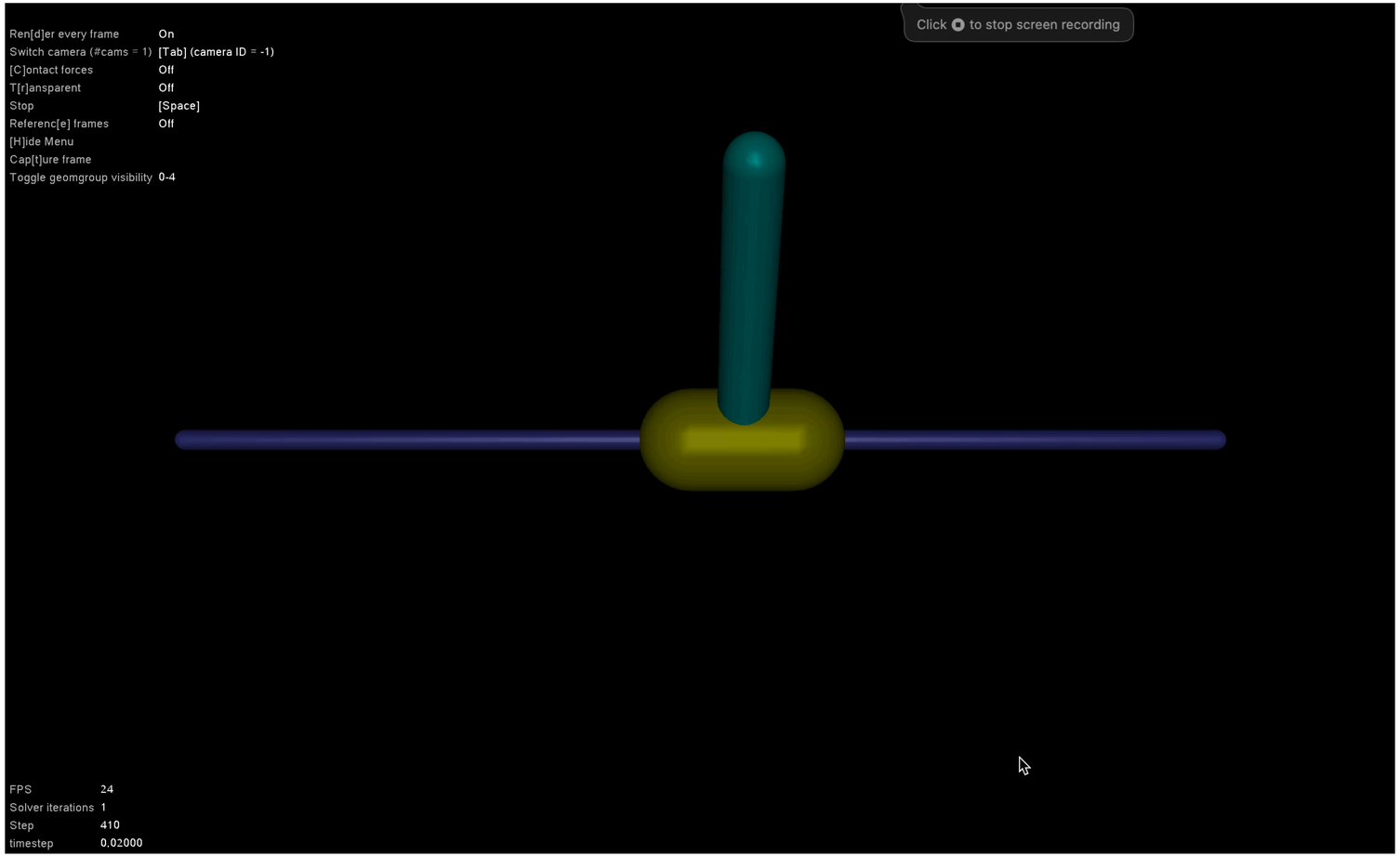
INVERTED PENDULUM

“InvertedPendulum-v5”



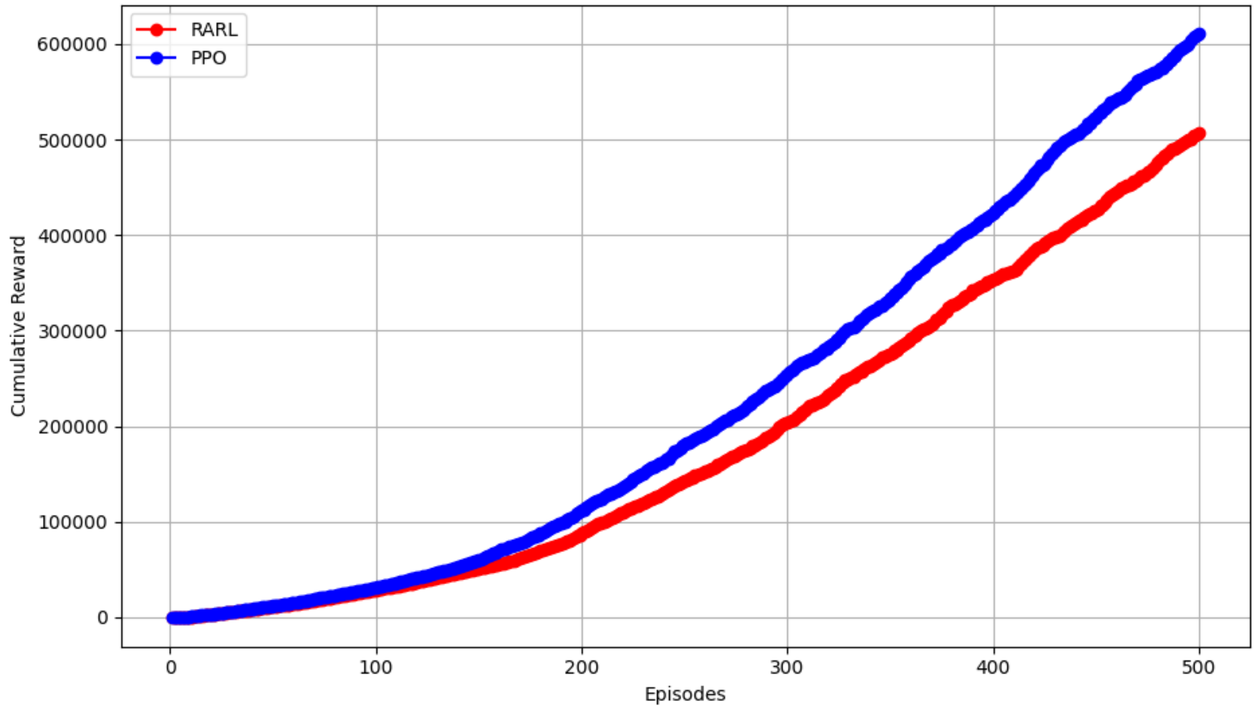
CUMULATIVE REWARD DURING EVALUATION

Inverted Pendulum



Walker2d

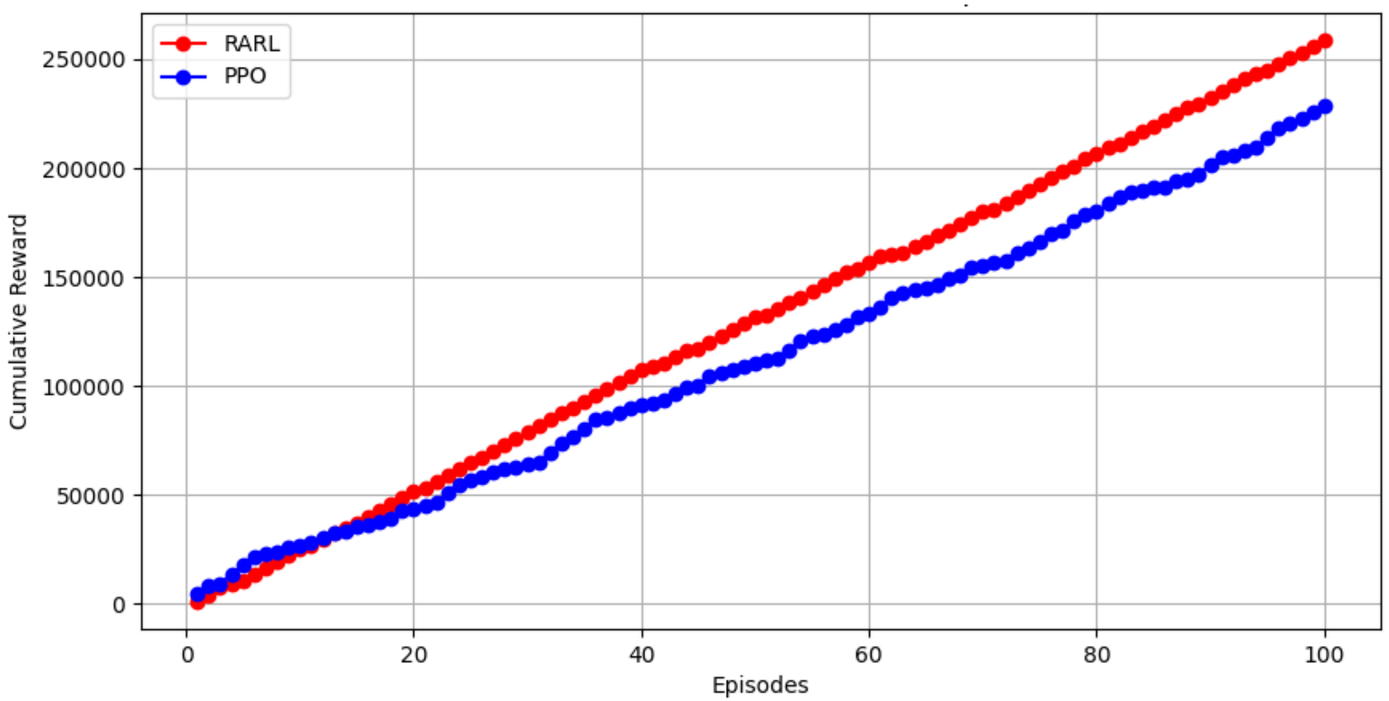
“Walker2d-v5”



CUMULATIVE REWARD DURING TRAINNING

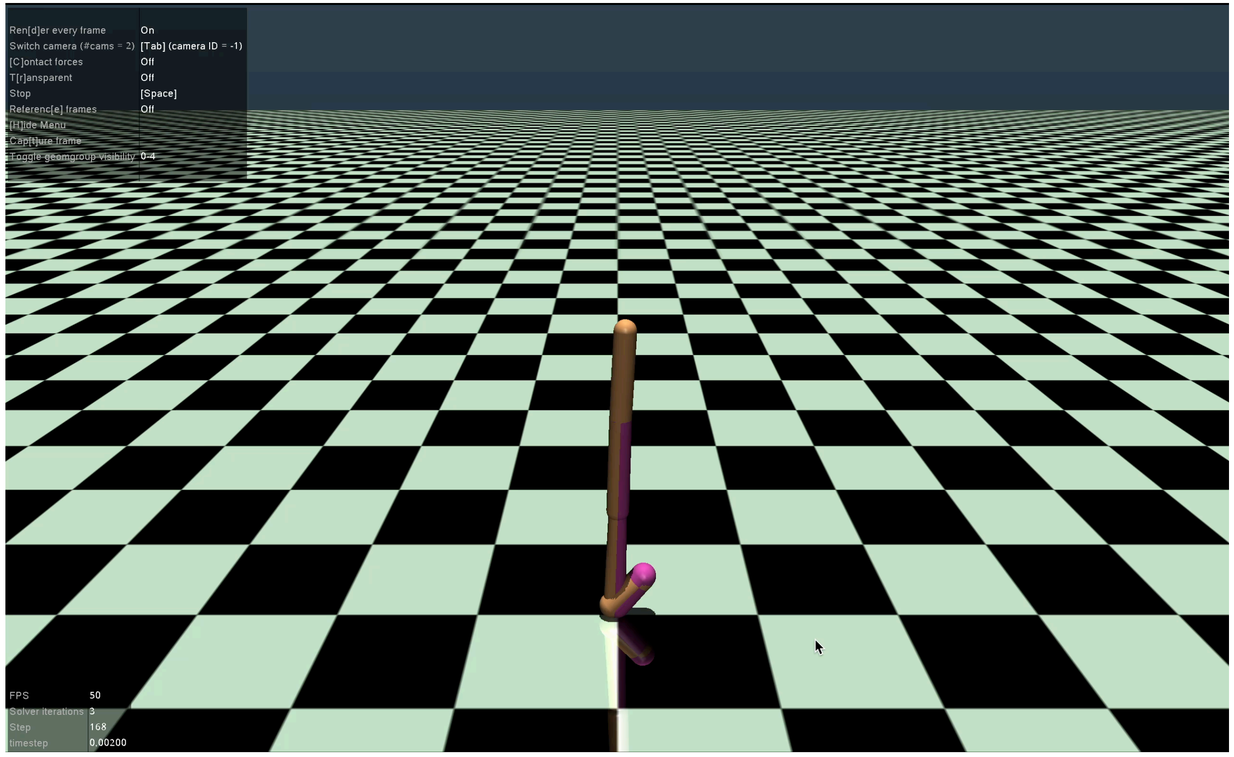
Walker2d

“Walker2d-v5”



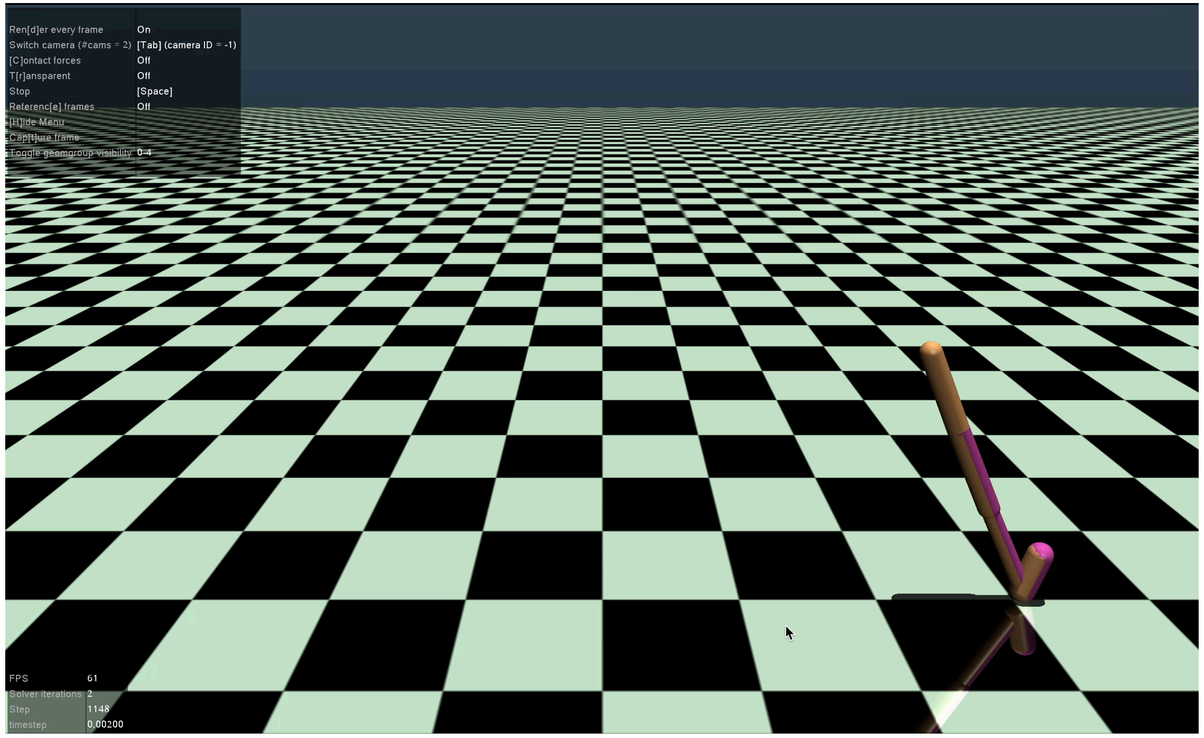
CUMULATIVE REWARD DURING EVALUATION

PPO-Baseline



N. TIMESTEPS: 50000

RARL 1



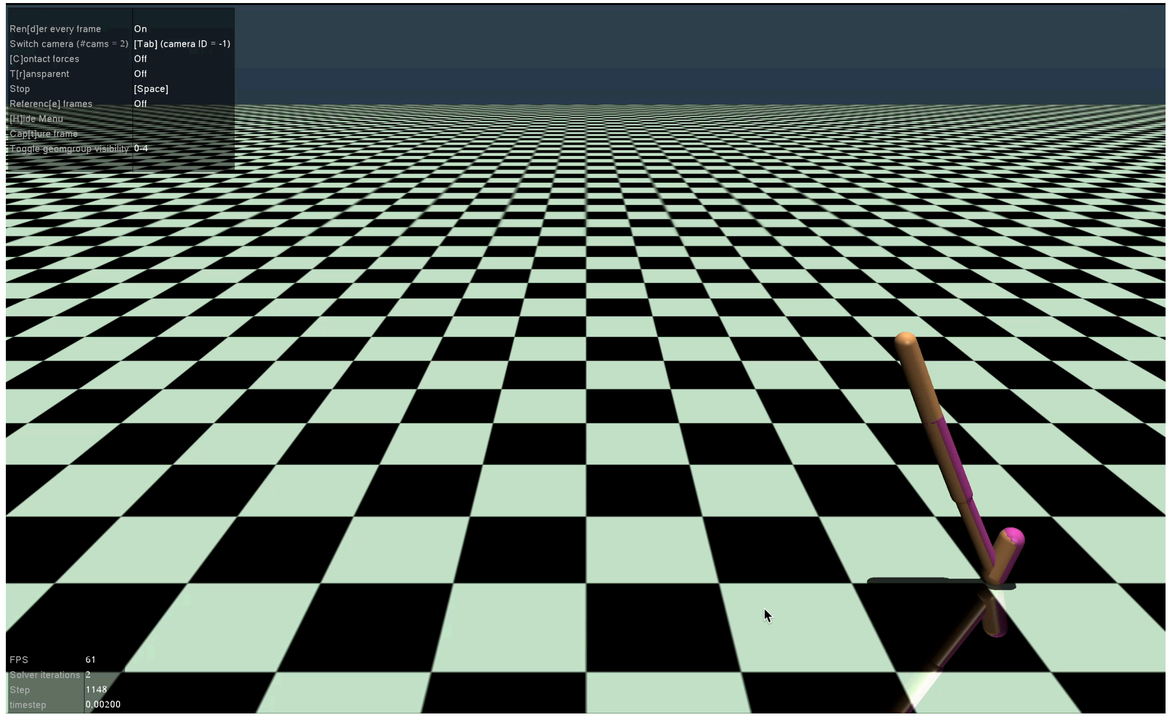
N. ITERATIONS: 500
N.PRO = N. ADV = 100

First Experiment

Improving Walker2d gait...

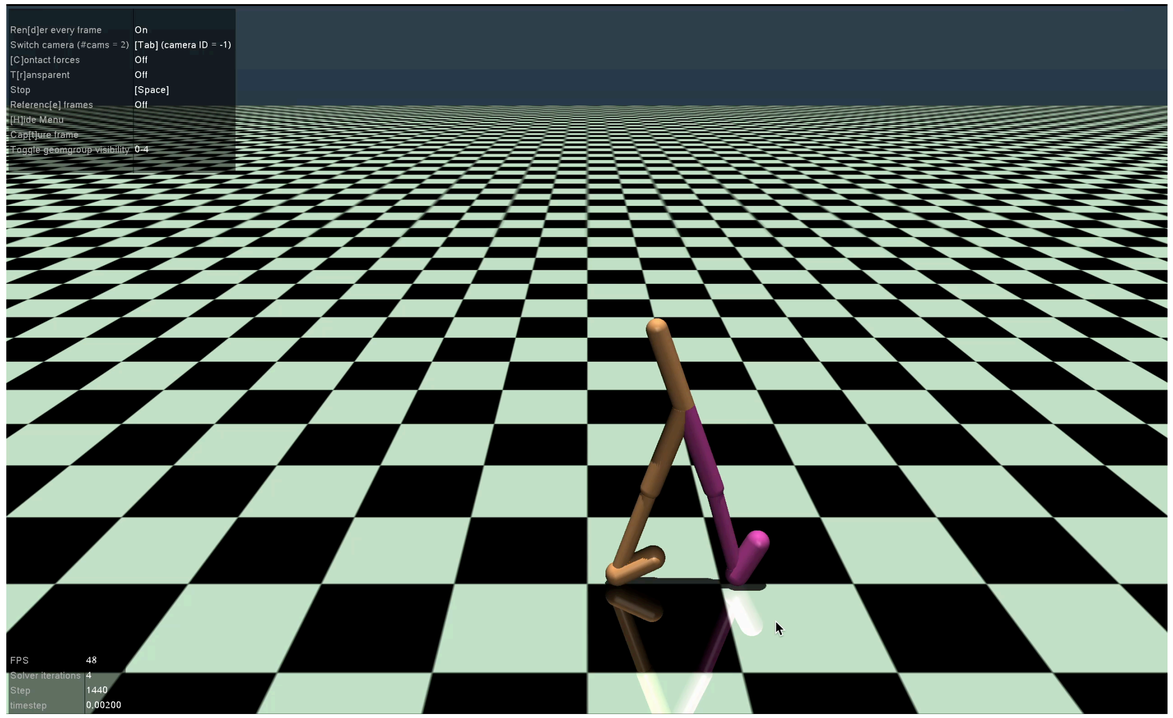
- Penalize when distant from “max velocity” (v_max= 1,5).
- Penalize when the agent is distant from a jumping threshold (z_th = 0.2).
- Penalize when the foot angles are distant from zero.
- Penalize if torso angle is distant from zero.
- Reward when the angular velocity of leg and thigh are higher than a certain threshold(w_th =0.3).

RARL 1



N. ITERATIONS: 500
N.PRO = N. ADV = 100

RARL 2



Higher control but lower rewards

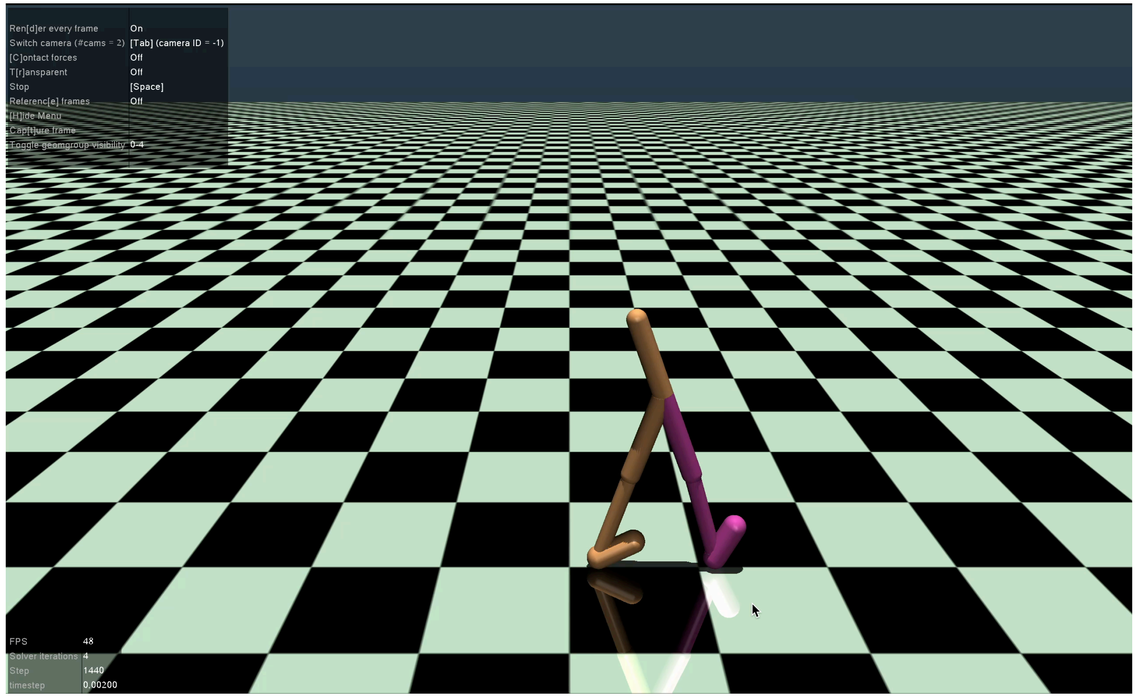
N. ITERATIONS: 500
N.PRO = N. ADV = 100

Second Experiment

Improving Walker2d gait...

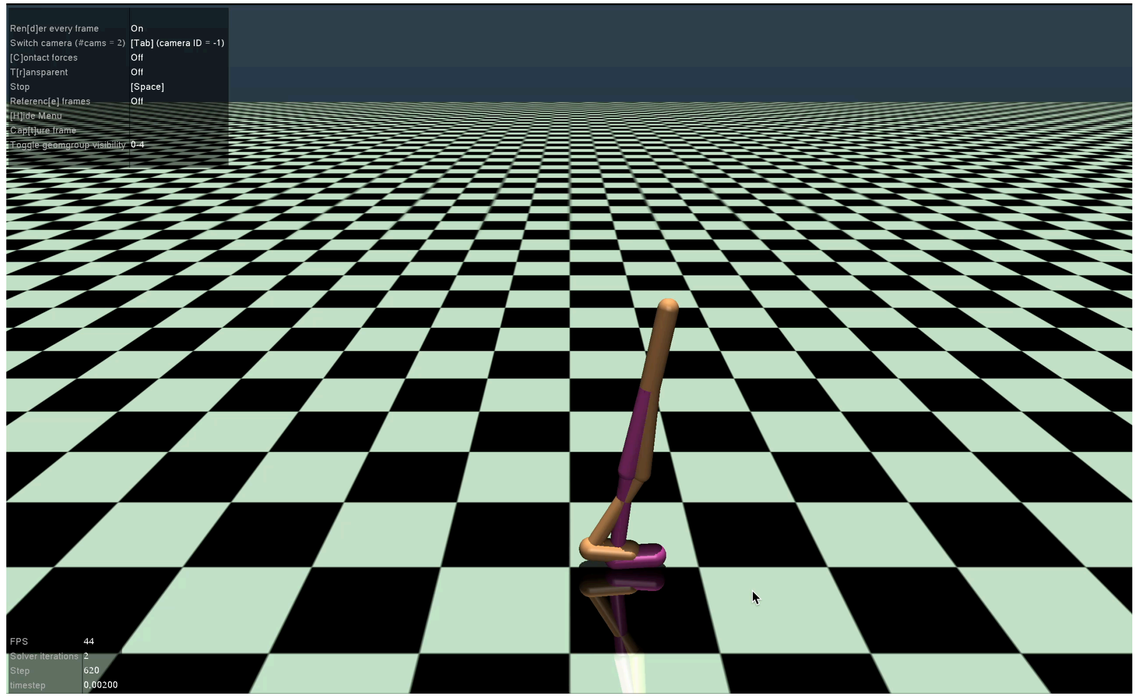
- Encouraging Sinusoidal movements.
- Rewarding higher difference between thighs angles.
- Rewarding opposite angular speed signs of thighs.
- Penalising too fast movements (jerk).
- Rewarding knees angular speeds.

RARL 2



N. ITERATIONS: 500
N.PRO = N. ADV = 100

RARL3 (best performance)



Periodicity -> stability, natural gait

N. ITERATIONS: 200
N.PRO = N. ADV = 150

Conclusion

- The Inverted Pendulum's mean reward follows a similar trend in PPO -Baseline and RARL, with a 200 iterations evaluation.
- The Walker2d agent trained with RARL demonstrates greater robustness compared to the PPO baseline.
- To achieve a more natural and stable gait, we found that incorporating a sinusoidal reward function along with appropriately fine-tuned parameters yields the best results.

Reference

- Robust Adversarial Reinforcement Learning: https://arxiv.org/abs/1703.02702