StableBaselines3 PPO and A2C Applied to the Highway-Env Racetrack

Aymane Lotfi[1], Abdelaziz Guelfane[1], Abdellah Oumida[1], and Idriss Mortadi[1]

[1]{aymane.lotfi,adbelaziz.guelfane, abdellah.oumida, idriss.mortadi}@student-cs.fr
GitHub Repository

April 23, 2025

Abstract

This report details the application and comparison of two prominent reinforcement learning algorithms, Proximal Policy Optimization (PPO) and Advantage Actor-Critic (A2C), for continuous lateral vehicle control in the `racetrack-v0` environment from the `highway-env` suite. Both agents were implemented and trained using the Stable-Baselines3 library, leveraging an `OccupancyGrid` observation space and parallel environment processing via `SubprocVecEnv`. The environment's reward structure was specifically tuned, including increasing the collision penalty to encourage safer driving. The primary focus is on the PPO agent, detailing its configuration, training dynamics, and evaluation. PPO achieved a mean evaluation reward of 843.72 ± 417.07 over 10 episodes. The A2C agent was also trained and evaluated, yielding a mean reward of 488.87 ± 604.81. While PPO demonstrated superior average performance, both agents exhibited significant performance variability, particularly A2C, underscoring the challenge of achieving consistent robustness in this task. Qualitative analysis of PPO showed competent lane-following but difficulties in complex multi-vehicle interactions. A direct comparison highlights PPO's more stable learning profile and better final results for this specific task configuration.

1 Environment and Setup

A consistent experimental setup is crucial for comparing RL algorithms. Both PPO and A2C agents were trained and evaluated using the following configuration, implemented via Stable-Baselines3 (SB3) and the `highway-env` suite.

1.1 Environment: `racetrack-v0`

The task involved navigating a closed-loop circuit with 3 other vehicles, using the `racetrack-v0` environment. The goal, guided by the reward structure, was safe and efficient lane keeping.

1.2 Environment Configuration

Key settings defined in `configs/task_3_config.py`:

- **Observation:** `"OccupancyGrid"` (12x12 cells, 3m resolution, 36x36m area, encoding presence/on-road status).
- **Action:** `"ContinuousAction"` (Lateral steering control only; longitudinal control disabled).
- **Reward Structure:** Penalty-driven.
  - Collision Penalty: `-3.5` (Increased from default to prioritize safety).
  - Lane Centering Cost: Factor 4 penalty based on deviation.
  - Action Penalty: `-0.3` for steering magnitude.
- **Vehicles:** 1 ego agent, 3 background vehicles.

1.3 Training and Evaluation Infrastructure

- **Library/Policy:** SB3 with `"MlpPolicy"`.
- **Parallelization:** `SubprocVecEnv` with `n_cpu = 6` (`task3_ppo.py`, `ac_plot_figs.py`).
- **Logging:** TensorBoard logs saved to `racetrack_ppo/` and `racetrack_a2c/`, plotted via helper scripts.
- **Evaluation:** Agents evaluated deterministically over 10 episodes using a script similar to `test_agent.py`, calculating mean/std reward and length. Videos were recorded for qualitative analysis (e.g., PPO videos saved via `RecordVideo`).

2 Proximal Policy Optimization (PPO)

2.1 Algorithm Overview

PPO is an on-policy actor-critic method enhancing stability by optimizing a clipped surrogate objective (Eq. 1) over multiple epochs per data batch, limiting policy changes.

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

Here, $r_t(\theta)$ is the policy probability ratio, $\hat{A}_t$ the advantage estimate, and ϵ the clipping parameter.

2.2 Training and Hyperparameters

The PPO agent was trained using `task3_ppo.py` for 400k timesteps. Key hyperparameters:

- `n_cpu`: 6
- `learning_rate`: `3e-4`
- `gamma`: `0.99`
- `n_steps`: `512`
- `batch_size`: `64`
- `total_timesteps`: `400k`

(SB3 defaults used for others like `n_epochs`, GAE λ, ϵ).

2.3 PPO Results and Analysis

2.3.1 Learning Progress (Training Curves)

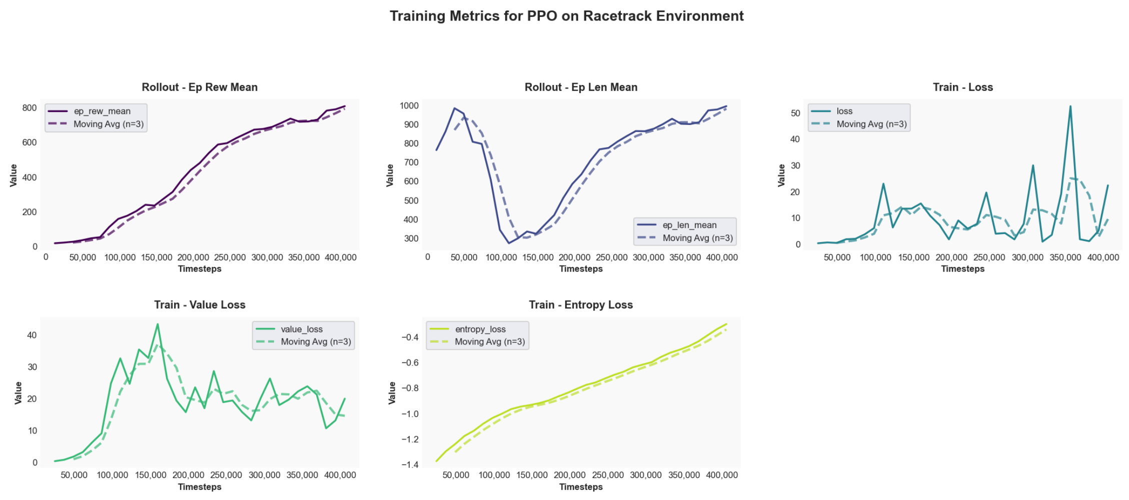


Figure 1: PPO Training Metrics on `racetrack-v0` (400k timesteps).

Figure 1 shows relatively stable learning: reward/length increase consistently (reaching 500-600 reward during training), entropy decreases (policy becomes deterministic), and losses stabilize. The value loss curve is notably smoother than A2C's.

2.3.2 Evaluation Performance

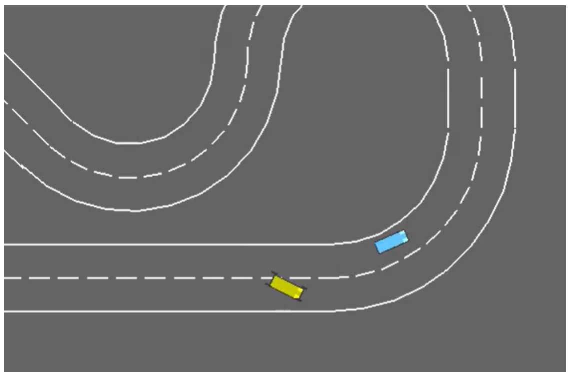
Quantitative evaluation over 10 deterministic episodes yielded:

- **Mean Reward:** 843.72 ± 417.07
- **Mean Length:** 1003.90 ± 497.15

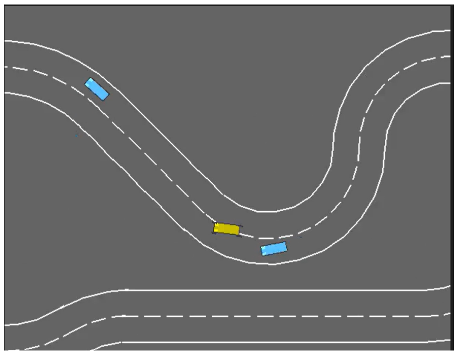
High average performance coupled with very high variance (50

2.3.3 Agent Behavior (Qualitative Analysis)

Videos confirm competent lane following (Fig. 2a) but show hesitation decisions in complex multi-vehicle scenarios (Fig. 2b), likely causing the high variance. Lateral-only control limits tactical options.



(a) Navigating near one vehicle.



(b) Near multiple vehicles.

Figure 2: Sample frames from PPO evaluation.

3 Advantage Actor-Critic (A2C)

3.1 Algorithm Overview

A2C is a synchronous, on-policy actor-critic algorithm. It updates policy/value networks using advantage estimates ($A(s, a) \approx R_t - V(s_t)$) after each batch of parallel rollouts, typically performing one update per batch without policy clipping.

3.2 Training and Hyperparameters

Trained using `ac_plot_figs.py` for 400k timesteps. Key hyperparameters differed from PPO:

- `n_cpu`: `6`
- `learning_rate`: `7e-4` (Higher)
- `gamma`: `0.99`
- `n_steps`: `5`
- `total_timesteps`: `400k`

(SB3 defaults for others). Small `n_steps` implies frequent updates on less data.

3.3 A2C Results and Analysis

3.3.1 Learning Progress (Training Curves)

Figure 3 shows A2C learned, but with issues: reward/length curves are highly volatile (peaking 450-500 reward), value loss is extremely unstable, and entropy loss increases anomalously.

3.3.2 Evaluation Performance

Quantitative evaluation over 10 deterministic episodes yielded:

- **Mean Reward:** 488.87 ± 604.81
- **Mean Length:** 524.20 ± 646.06

The A2C agent achieved a significantly lower average reward compared to PPO. Furthermore, the standard deviation is extremely high, even exceeding the mean reward ($\approx 124\%$ std dev relative to the mean), indicating extreme inconsistency and poor robustness. Videos were saved to monitor the agent behavior.

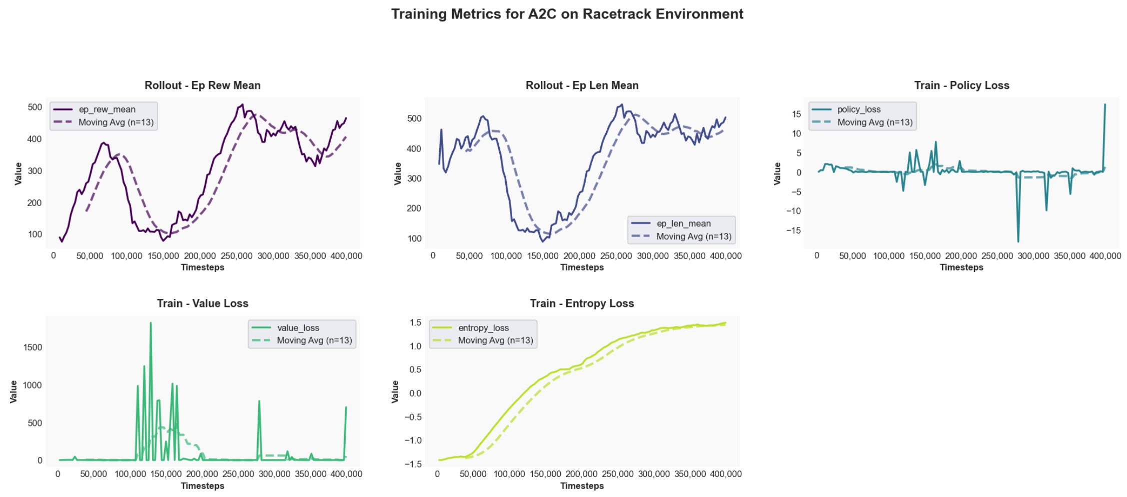


Figure 3: A2C Training Metrics on `racetrack-v0` (400k timesteps).

4 Comparison: PPO vs. A2C

Key differences observed in this experiment:

- **Stability:** PPO learning curves were significantly more stable than A2C's erratic training dynamics.
- **Training Performance:** PPO achieved slightly higher and more stable peak rewards during training.
- **Convergence Indicators:** PPO's decreasing entropy suggests better convergence than A2C's anomalous increasing entropy.
- **Setup Differences:** PPO's larger `n_steps` (512 vs 5), lower learning rate, and internal mechanisms (clipping, multiple epochs) likely contribute to its stability advantage over A2C's frequent, small-batch updates here. The tuned collision reward (-3.5) was applied to both training setups.
- **Final Performance:** PPO achieved a substantially higher mean evaluation reward (843.72) compared to A2C (488.87). While both showed high variance, A2C's variance was exceptionally large relative to its mean, confirming its poor robustness suggested by the unstable training.

For this setup, PPO demonstrated superior training stability and delivered significantly better, albeit still inconsistent, final performance compared to A2C.