Reinforcement Learning Report
– CarRacing-v3 –

by
Arnaud Fernandes, Valentin Gorse, Yacine Lassouani

1. Introduction & Problem Overview

This project aimed to design and train a reinforcement learning agent capable of autonomous driving in a simulated racing environment.
We chose CarRacing-v3 from Gymnasium, a classic benchmark where an agent must drive around procedurally generated tracks using only raw pixel observations.

The core challenge lies in continuous control. The agent must coordinate steering, throttle, and braking based solely on visual feedback, with no prior knowledge of vehicle dynamics or racing rules.
The task tests an agent's ability to balance speed, stability, and anticipation while adapting to random track layouts.

To tackle this, we implemented and trained a Proximal Policy Optimization (PPO) agent capable of directly mapping image sequences to smooth continuous actions.
Our objective was to develop a policy that could consistently complete laps with human-like precision and efficiency.

Finally, to make the project interactive and engaging, we implemented a "Human vs AI" mode. A split-screen race where a human player can directly compete against the trained agent.

2. Environment & Reward Function

Link: https://gymnasium.farama.org/environments/box2d/car_racing/

The CarRacing-v3 environment from Gymnasium provides a top-down 2D view of a car driving on procedurally generated racing tracks.
At each timestep, the agent receives an RGB image of size 96×96 pixels, representing its current field of view.
The goal is to complete as much of the track as possible while minimizing time off the road.

Action Space (Discrete):

- 0: do nothing
- 1: steer right
- 2: steer left
- 3: gas
- 4: brake

Action Space (Continuous):

The control is continuous, defined by a 3-dimensional action vector:

- Steering ∈ [-1, 1]: controls left and right turns.
- Gas ∈ [0, 1]: accelerates the car forward.
- Brake ∈ [0, 1]: slows the car down.

This continuous formulation allows smoother and more realistic driving behaviors compared to discrete control, avoiding abrupt directional changes.

Observation Space:

The observation consists of pixel-based visual input, which is then:

- Converted to grayscale to reduce dimensionality.
- Stacked over 4 consecutive frames to capture short-term temporal dynamics (velocity, direction, etc.).

This gives the agent a richer representation of motion without requiring explicit state information.

Reward Function:

The agent gains a positive reward for every new section of track visited and loses points when it slows down or leaves the road.
Episodes end early when the car stops making progress, encouraging steady forward motion and penalizing instability.

3. Algorithm & Training Setup

We used Proximal Policy Optimization (PPO) implemented in PyTorch.
After testing a discrete-action PPO version, which reached high scores but suffered from large variance (up to -120 after +900 runs), we switched to continuous control, which offered smoother steering and better stability.

Model Architecture:

- Convolutional neural network (CNN) encoder for visual input.
- Two heads: a Gaussian policy (actor) and a scalar value function (critic).
- Optimization with Adam, using GAE-lambda for stable advantage estimation.

Training Strategy:

To achieve robust and stable learning, we designed a multi-phase training schedule over 6 million environment steps.

Each phase progressively adjusted the learning rate, entropy coefficient, and rollout horizon, the three key hyperparameters driving PPO's balance between exploration and exploitation.

Phase	Steps	Learning Rate	Entropy Coef	Rollout Horizon
1	0-1.5M	5e-5	1e-3	4096
2	1.5-3M	3e-5	7e-4	3072
3	3-4.5M	1.5e-5	4e-4	2048
4	4.5-6M	5e-6	1e-4	1024

Hyperparameter explanations:

- Learning Rate (LR):

Controls how much the model's parameters are updated at each step.
A lower LR (like 5e-5 instead of 1e-3) helps prevent instability and overshooting, which is common in high-dimensional visual RL tasks.
We start with a small value to ensure gradual, stable learning, and decrease it further to fine-tune the model in later stages.

- Entropy Coefficient:

Encourages exploration by adding randomness to the policy.
High entropy (1e-3) at the start allows the agent to test diverse actions, while gradually reducing it helps stabilize and refine its behavior once good strategies are found.

- Rollout Horizon:

Determines how many environment steps are collected before each policy update.
Larger horizons (4096) capture long-term dependencies early in training, whereas shorter ones later (1024) make learning updates faster and more precise once behavior stabilizes.

This gradual annealing strategy proved essential to balance exploration and exploitation. The agent explored broadly during early training, then focused on perfecting its trajectory and control precision as learning progressed.

Monitoring and Checkpoints:

All training runs were logged in real time using a custom visualization script (live_plot.py), which plotted:

- Episode rewards
- Moving averages
- Learning rate decay

Each time the agent reached a new record in average reward, its weights were automatically checkpointed, ensuring no progress was lost.

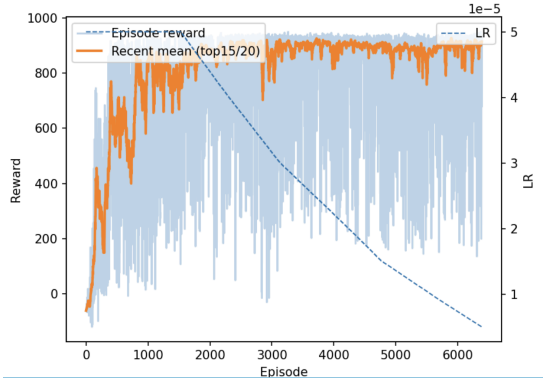
Quick view on different checkpoints:

Before reaching the final version, we experimented with several architectures and configurations, but this combination of adaptive scheduling and consistent monitoring led to stable and reproducible performance improvements throughout the six million training steps.

4. Experiments & Results

After six million training steps, the continuous PPO agent achieved average rewards around 900, with peaks near 950.
Performance became more stable over time as entropy decreased, reflecting the agent's shift from exploration to mastery.

The reward curve showed a steady improvement, with the moving average staying above 900 during late training.
Failures, typically when the car spun off-road, became rare after phase 3.



During evaluation, the agent demonstrated:

- Smooth and anticipatory control, braking before turns and accelerating out of corners.
- Consistent lap completion even on unseen tracks.
- Human-like stability, avoiding erratic steering typical of earlier versions.

Typical late-stage log:

```
[final4 ep 6378] rew=884.08 recent_mean=910.12 steps=5990074 lr=5.1e-06 ent=0.0001 horz=1024 phase=3
[final4 ep 6379] rew=867.85 recent_mean=905.21 steps=5991074 lr=5.1e-06 ent=0.0001 horz=1024 phase=3
[final4 ep 6380] rew=900.90 recent_mean=906.83 steps=5992065 lr=5.1e-06 ent=0.0001 horz=1024 phase=3
[final4 ep 6381] rew=200.00 recent_mean=906.83 steps=5993065 lr=5.1e-06 ent=0.0001 horz=1024 phase=3
[final4 ep 6382] rew=877.85 recent_mean=903.87 steps=5994065 lr=5e-06 ent=0.0001 horz=1024 phase=3
[final4 ep 6383] rew=934.70 recent_mean=908.59 steps=5994718 lr=5e-06 ent=0.0001 horz=1024 phase=3
[final4 ep 6384] rew=890.85 recent_mean=910.12 steps=5995718 lr=5e-06 ent=0.0001 horz=1024 phase=3
[final4 ep 6385] rew=928.00 recent_mean=913.47 steps=5996438 lr=5e-06 ent=0.0001 horz=1024 phase=3
[final4 ep 6386] rew=919.40 recent_mean=913.21 steps=5997244 lr=5e-06 ent=0.0001 horz=1024 phase=3
[final4 ep 6387] rew=677.78 recent_mean=911.45 steps=5998244 lr=5e-06 ent=0.0001 horz=1024 phase=3
[final4 ep 6388] rew=919.50 recent_mean=913.83 steps=5999049 lr=5e-06 ent=0.0001 horz=1024 phase=3
🏁 FINAL PHASE CONTINUOUS TRAIN DONE
```

5. Analysis & Conclusion

The final PPO model achieved stable and generalizable driving behavior:

- **Consistency:** Regular scores above 900 demonstrated strong policy stability.
- **Realism:** Continuous actions led to natural driving behavior with fine control.
- **Generalization:** The policy adapted well to unseen tracks, not just memorized layouts.
- **Limitation:** Recovery from off-road situations remains weak due to PPO's lack of memory.

The 4-phase scheduler proved critical: it kept early exploration broad and gradually locked in optimal behavior, resulting in smoother driving and faster convergence.

The Human vs AI mode added a creative dimension. A split-screen race where the trained agent competes directly with a player.
The AI generally outperformed casual players, showcasing learned precision and control while keeping the game engaging and competitive.

In conclusion, this project successfully demonstrated how reinforcement learning can produce a reliable and adaptable autonomous driver using only visual input.
Through careful tuning of PPO and progressive learning schedules, the model achieved human-level stability and control, and even became a worthy opponent on the virtual racetrack!

6. Team & Roles

This CarRacing project started as a side experiment while our main focus was on the Pong environment.
Originally meant as a backup, it became a successful and entertaining project demonstration.

- Arnaud Fernandes – Reinforcement Learning Lead

Designed and implemented the CarRacing PPO agent, tuned hyperparameters, and managed the 6M-step training schedule.
Also developed the evaluation and Human vs AI racing interface.

- Yacine Lassouani – Experimental Support & Debugging

Assisted in fine-tuning, optimizing performance parameters, and analyzing training behavior.
Helped ensure stable training and consistent evaluation runs.

- Valentin Gorse – Documentation & Presentation

Proposed design ideas for the interactive racing demo, structured the report, and prepared the visual presentation for the final defense.

Together, we combined research, coding, and creativity to develop a fully autonomous racing agent. One that can drive, learn, and even compete against humans.