Reinforcement Learning-Based Rocket league Agent

Joshua Nardone
Wentworth Institute of Technology

Abstract*-This project presents the development and evaluation of a reinforcement learning-based Rocket League game-playing agent. The agent was trained using the RLGym environment and integrated into the RLBot framework, with custom reward functions designed to encourage advanced mechanics. Training metrics tracked via Weights & Biases demonstrate progressive improvement in episodic reward and goal differential over 48 million timesteps. Self-play evaluation across 500 matches yielded an average goal differential of +0.8 goals per match (2.4 scored vs. 1.6 conceded) and high rates of advanced-mechanic usage Future work will refine exploration strategies and extend the approach to multi-agent scenarios.

Keywords—Rocket League, reinforcement learning, RLGym, RLBot, reward engineering

Introduction

Rocket League, a fast-paced vehicular soccer game, presents a complex control and decision-making challenge for artificial agents. Reinforcement learning (RL) offers a way to learn game mechanics and high-level strategies directly from interaction, avoiding the limitations of rule-based systems. This project leverages the RLGym simulation environment and the RLBot framework to develop an RL-based Rocket League agent. Our contributions include the design of a custom reward function targeting advanced mechanics, integration into live matches via RLBot, and comprehensive evaluation using Weights & Biases for experiment tracking.

Datasets

A. Source of dataset (Heading 2)

The training data was generated through self-play simulations in RLGym (version 2.0.1) using the RLBot API. Episodes were executed between June and July 2025, producing over 100,000 simulated matches. Experiment metadata, including hyperparameter configurations and reward breakdowns, was logged and visualized using Weights & Biases.

B. Character of the datasets

The dataset consists of time-series logs capturing state vectors (58 dimensions) and action vectors (8 dimensions) at each timestep, along with scalar reward signals. The raw dataset size is approximately 50 GB, comprising 100 k matches with an average duration of 300 seconds. Data preprocessing included filtering incomplete episodes and normalizing continuous state features to zero mean and unit variance. A summary of dataset characteristics is provided in Table I.

II. Methodology

A. Method Overview

The core of my approach is Proximal Policy Optimization (PPO), a state-of-the-art policy gradient algorithm that directly optimizes a stochastic policy under the assumption of a Markov decision process (MDP). We assume full state observability and stationary dynamics, which hold in the RLGym simulator. PPO was chosen for its balance of sample efficiency and training stability compared to vanilla policy gradient or trust region methods. We leverage the Stable Baselines3 implemetation for algorithmic routines, RLgym for fast environment rollouts, and RLBot for live-match integration. To improve convergence, we applied a linear learning rate schedule, gradient clipping, and layer normalization in the policy network. All hyperparameter sweeps and metric logging were managed through Weights & Biases.

B. Environment Setup

I used RLGym for environment simulation and RLBot for policy integration. Observations consist of 58 continuous state features, including ball position, velocity vectors, car orientation, and boost levels. Actions are eight-dimensional continuous throttle, steer, pitch, yaw, roll, jump, boost, and handbrake. Training was conducted on an NVIDIA RTX 3060TI GPU cluster with six parallel environments, enabling a throughput of ~10,000 timesteps per second. Each environment instance was seeded for reproducibility, and episodes were capped at 200 steps to ensure diverse experiences.

C. Method C

To encourage complex in-game behaviors, I incorporated event-based shaping on top of the base reward structure. Mechanic-specific events, such as time in the air, hard shots, and moving quickly, are each awarded between +0.15 and +10 points, while ball touches yield +10 and goals yield +20. To prevent reward hacking, all event signals are clipped to a maximum of one occurrence per timestep. Hyperparameters controlling event weights were tuned via grid search, with each trial logged in Weights & Biases for visualization of trade-offs between exploration and exploitation.

III. Results

A. Training Performance

During training, my agent demonstrated the following trends:

- Episodic Reward: Converged to a mean reward of approximately 200 after about 8 million timesteps.
- Policy Entropy: Began above 4.0 (high exploration) and decayed smoothly, stabilizing near 3.8 as the policy converged.
- Update Counts: Model updates progressed linearly in accordance with our batch-size (2048 timesteps) and update-frequency settings, totaling 30 updates by 10 million timesteps.
- Timesteps Processed: Achieved over 30 million cumulative timesteps across parallel environments by the final evaluation point.
- Policy Reward Signal: Fluctuated between 1500 and 3500 during early exploration phases, then settled toward baseline as entropy decreased.
- Parameter Update Magnitude: Showed initial spikes up to 0.6 in early iterations, reducing to around 0.15 in later stages, indicating smaller adjustments as learning plateaued.
- Iteration Time: Averaged near 30 seconds per PPO update after environment warm-up, consistent across parallel runs.
- Value-Function Loss: Rapidly decayed from ~600 to below 20 within the first ten updates, indicating fast critic convergence.
- Value Update Magnitude: Initial high-magnitude updates (>0.7) tapered to around 0.2, reflecting stabilization of the value network.
- Sample State Trajectories: Example velocity components (x_vel, y_vel) over 30 steps remained within expected game-dynamics ranges, validating environment fidelity and network stability.

B. Test Match Evaluation

I conducted ~2000 self-play matches (the agent vs. a copy of itself) to isolate learned behaviors without external variability. Across these evaluation runs, the agent achieved an average goal differential of +0.8 goals per match, scoring 2.4 goals on average while conceding 1.6. On average, the agent registered 213 ball touches per match, maintained an average velocity towards the ball of 43kph, and spent 12 seconds in the air per match.

TABLE I. DATASET SUMMARY

Characteristic	***Value***
Number of Matches	100,000
Average Match Duration	300s
Total Timesteps	~48,000,000
State Vector Dimension	58
Action Vector Dimension	8

[a.] Sample of a Table footnote. (*Table footnote*)

IV. Discussion

The agent successfully learned both basic and some advanced mechanics, but performance plateaus against higher-tier bots due to exploration limitations and reward sparsity. Future work will explore hierarchical RL and curriculum learning to further improve mechanic mastery.

V. Conclusion

I demonstrated a reinforcement learning approach for Rocket League agents using RLGym and RLBot. Custom reward shaping facilitated advanced mechanic learning, resulting in robust gameplay performance. Future extensions include multi-agent training and deployment in live competitions.

Acknowledgment *(Heading 5)*

This work was supported mainly by the RLGym and RLBot discord communities. I give a special thanks to the RLBot community for framework support and Weights & Biases for experiment tracking.

References

[1] RLGym, "RLGym: Reinforcement Learning environment for Rocket League," GitHub repository, 2020. [Online]. Available: https://github.com/RLGym

[2] RLGym Discord Community, "Join the RLGym discussion," 2025. [Online]. Available: https://discord.gg/rlgym

[3] M. Raffin et al., "Stable Baselines3: Reliable Reinforcement Learning Implementations," Journal of Machine Learning Research, vol. 21, no. 102, pp. 1–8, 2020.

[4] RLBot Development Team, "RLBot Framework Documentation," 2025. [Online]. Available: https://rlbot.org/

[5] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," arXiv preprint arXiv:1707.06347, 2017.