Maxim Loukine (101275222)
COMP3106 - Final Report
Group 122

Reinforcement Learning Rocket League Bot

1. Introduction

1.1 Background, Motivation, and Objectives

Rocket League is a fast-paced 3-D physics world where rocket-powered cars play Soccar by hitting a ball into a goal. Soccer in general can be alternatively named Soccar in the context of Rocket League. Unlike traditional reinforcement learning tasks that involve environments like gridworld or Atari games, Rocket League provides a challenging scenario. In Rocket League, the world is a continuous one, having complex car physics. Additionally, the world contains partial observability and also offers sparse rewards for tasks.

The motivation for this project was to construct a compact and interpretable baseline that demonstrates how tabular Q-learning behaves in such a demanding environment. Rather than pursuing performance with deep neural networks as many of the successful projects in machine learning bots in Rocket League, the focus here is on explainability and reproducibility. By reducing the observation space to two dimensions, distance to the ball and car speed and restricting the action set to a small discrete subset we can clearly observe how discretization, reward shaping and exploration interact to produce learning. This design allows us to surface the practical limits of tabular Q-learning in Rocket League and to provide a foundation for future work that scales to more sophisticated methods.

First, to develop a discrete-state Q-learning agent to touch the ball to prove that sparse rewards can also be a driving factor when shaping is used. Second, to ensure a reproducible per/episode and per/run output of logs for the analysis of the proposed approach. Third, to analyze the results obtained in the experiment to present the points where the approach has been successful. Finally, to recommend the next steps to be taken for further improvements.

1.2 Related Prior Work

The theoretical foundation of this project lies in tabular Q-learning, introduced by Watkins and Dayan. They describe Q-learning as “a simple way for agents to learn how to act optimally in controlled Markovian domains” and prove that the algorithm converges to optimal action values with probability one, provided that rewards are bounded, learning rates satisfy certain conditions, and all state-action pairs are repeatedly sampled (Watkins & Dayan, 1992, pg. 279). This convergence theorem provides the baseline against which this Rocket League experiment can be evaluated.

In this case, the assumptions required for convergence are difficult to satisfy in continuous, high dimensional domains like Rocket League. Recent research has therefore focused on adapting Q-learning to continuous state-action spaces through discretization and abstraction. A 2024 study introduced symbolic abstraction methods that generate minimal and maximal Q-tables to bound Q-values in continuous domains, showing that finer discretization levels tighten these bounds and improve convergence. Other approaches include uniform grid discretization, Voronoi partitions, and adaptive space partitioning, each attempting to balance tractability with fidelity to the underlying dynamics.

In practice, scalable reinforcement learning in Rocket League has relied on deep methods. Deep Q-Networks (Mnih et al., 2015) demonstrated human level control in Atari games by combining Q-learning with convolutional neural networks. Policy gradient methods such as Proximal Policy Optimization (Schulman et al., 2017) have become standard for continuous control tasks, offering stability and scalability that tabular methods cannot achieve. These approaches highlight the trajectory of reinforcement learning research: while tabular Q-learning remains valuable as a pedagogical baseline, deep reinforcement learning is necessary for competitive performance in complex environments like Rocket League.

2. Methods

2.1 Algorithm and Action Selection

The agent was trained using tabular Q-learning, implemented in the project's qlearning.py. The Bellman update rule was applied after every transition:

$$Q(s,a) \leftarrow Q(s,a) + a[r + \gamma max_{a'} Q(s',a') - Q(s,a)]$$

This update is the standard off policy temporal difference control algorithm. As Sutton and Barto explain, "Q-learning is an off-policy TD control algorithm" (2018, pg. 149). The method updates values as if following the greedy policy, regardless of the behavior policy actually used. This property allows Q-learning to converge toward optimal action values even when exploration deviates from the target policy.

Exploration was managed through an ε-greedy policy with per episode decay. To boost sample efficiency in the continuous control setup of a game like Rocket League, a steering-aware selector has been incorporated. The selector makes use of geometric metrics including the angle to the ball and the distance to the target. Why the online learning approach was chosen. There

was a need to boost sample efficiency in the continuous control setup of a game like Rocket League. During exploitation, the agent selects the action that maximizes Q for the current state.

2.2 Observation Discretization

Observations came from the DefaultObs builder in RLGym. These normalized values were unnormalized by the scales of the observation space (SIDE_WALL_X, BACK_NET_Y, CEILING_Z) and the maximum speed of the car. Subsequently, the discretizer calculated the Euclidean distance to the ball and the car's speed magnitude. These values were put into discrete bins via floor division. The bins had typical sizes of 100 units for distance values and 100 units/sec for speed values. Such resulted in a grid of approximately 106-119 bins for distances and 27-29 bins for speed.

Such a method corresponds to the general problem of discretization in continuous settings. Indeed, as recently shown in the literature, uniform discretization may lead to the problem of "reachable mismatch," where the paths intersect the boundaries of the cells instead of being perfectly around the center of the cells (Alaoui & Saoud, 2024, Section 3).

2.3 Action Space Reduction

The entire action space of the game was reduced to four discrete actions (forward left, forward, forward right, boost forward) by a lookup table supplied by RLGym. This enabled the application of the algorithm of tabular Q-learning to the problem of selecting steering-aware actions.

2.4 Reward Shaping

Rewards shaping was a very important technique in dealing with the sparse reward environment of the game. The strong reward signal was the TouchReward. Dense rewards signals in the environment include SpeedTowardBallReward, SteeringTowardBallReward, FacingBallReward, BallToGoalReward, and BoostAlignmentReward. These signals are weighed to form the overall reward.

This approach to reward shaping aligns well with the explanation about shaping and eligibility traces in Sutton & Barto (2018, Ch. 7) Their approach helps to speed up the learning process by issuing mid-course corrections. In our case, the reward shaping helped to reduce the circling action while boost utilization was promoted by the sparse touch reward.

2.5 Training Protocol

Training runs consisted of 1,000–2,000 episodes, each up to 5,000 steps. Hyperparameters included a learning rate $\alpha = 0.2$, discount factor $\gamma = 0.99$, and ε initialized at 1.0 with per-episode decay ≈ 0.9965 until reaching a minimum of 0.02.

2.6 Logging

Per episode statistics (touches, average distance, alignment, forward percentage, and episode return) were logged to data/episode_stats.csv. Per run summaries (timestamp, states discovered, grid dimensions, sparsity, max Q, mean Q, close range avg Q) were appended to data/training_metrics.csv. Visualizations included Q-table heatmaps (max Q, avg Q, visitation).

3. Results

3.1 Qualitative Results

There was a qualitative continuity in the results of the training. The initial rounds consisted of random wandering, where the contact between the ball and the agent was minimal. Over a period of time, the onset of goal-oriented ball seeking was noticed in the agent. This was in terms of turning to the ball directionally, building speed when close to the ball, touching the ball intermittently.

The trend in the above figures can be understood best in the very first plot of the number of touches per episode. In that sequence of the experiment, the number of touches was very low, occurring approximately every 30 episodes. This underlines the challenges of learning for rewards in Rocket League.

Maxim Loukine (101275222)
COMP3106 - Final Report
Group 122

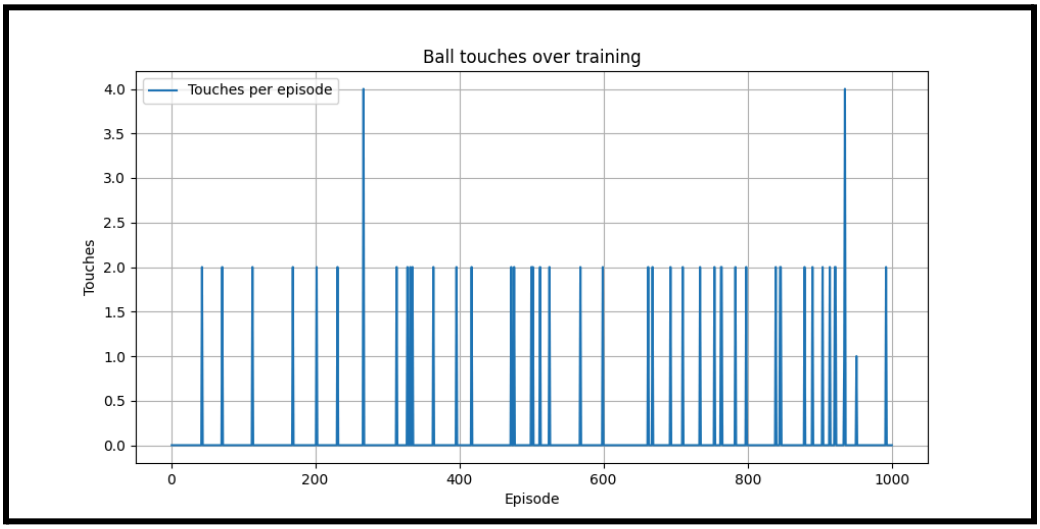

Figure 3.1: Early training run showing touches per episode. Touches occur approximately once every 30 episodes.

Later runs showed more frequent touches and upward trends in moving averages. Q-table visualizations confirmed that maximum Q-values concentrated in states with small distance buckets and low to moderate speed buckets. Average Q values spread more broadly across midrange speed buckets, reflecting the contribution of dense shaping rewards. Visitation maps revealed that exploration was concentrated in mid distance regions, where ball and car interactions occurred most often.

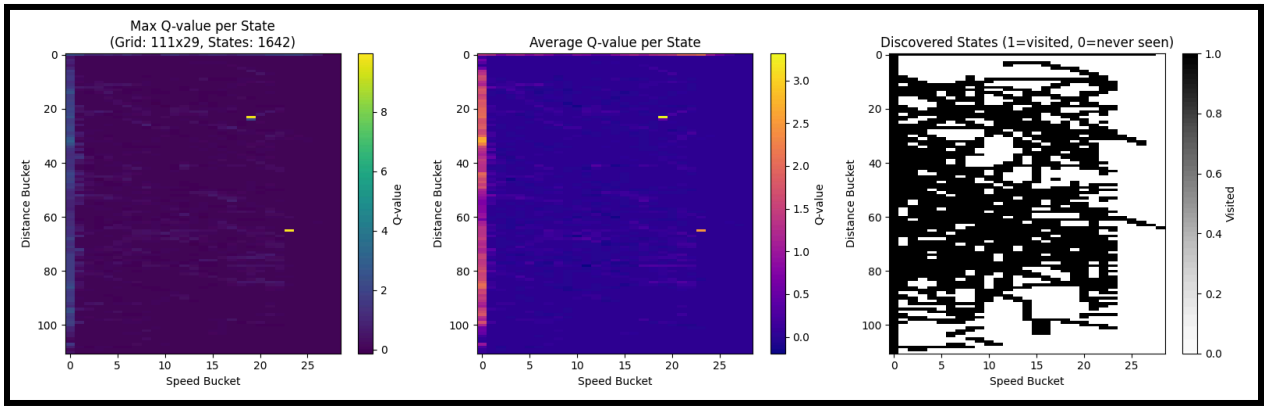

Figure 3.2: Q-table visualization from a 2000 episode run. Max Q values concentrate near close distance states, average Q spreads across midrange speeds, and visitation shows broad exploration with 51% grid coverage.

3.2 Quantitative Results

Quantitative metrics provide further evidence of learning. Summarized runs from training_metrics.csv

- Run 1 (1000 episodes): 932 states discovered; grid 106 × 28; sparsity 31.4%; max Q = 8.82; mean Q = 0.33; close range avgQ = 2.29.
- Run 2 (1000 episodes): 1226 states discovered; grid 107 × 29; sparsity 39.5%; max Q = 3.26; mean Q = 0.25; close range avgQ = 1.56.
- Run 3 (2000 episodes): 1252 states discovered; grid 111 × 28; sparsity 40.3%; max Q = 9.95; mean Q = 0.24; close range avgQ = 1.28.
- Run 4 (2000 episodes): 1408 states discovered; grid 110 × 29; sparsity 44.1%; max Q = 7.88; mean Q = 0.22; close range avgQ = 1.36.
- Run 5 (2000 episodes): 1277 states discovered; grid 114 × 27; sparsity 41.5%; max Q = 7.95; mean Q = 0.27; close range avgQ = 1.65.
- Run 6 (2000 episodes): 1642 states discovered; grid 111 × 29; sparsity 51.0%; max Q = 9.96; mean Q = 0.22; close range avgQ = 1.34.
- Run 7 (2000 episodes): 1113 states discovered; grid 117 × 29; sparsity 32.8%; max Q = 10.09; mean Q = 0.30; close range avgQ = 1.44.
- Run 8 (2000 episodes): 1172 states discovered; grid 111 × 28; sparsity 37.7%; max Q = 6.46; mean Q = 0.25; close range avgQ = 1.23.

This data shows that training runs typically explored 900–1600 unique states out of approx. 3000 possible grid cells, yielding sparsity levels of 30–51%. Maximum Q-values ranged from 3.26 to 10.3, suggesting that in successful runs the agent discovered action sequences with substantial expected return. Mean Q-values remained low (0.22–0.33), reflecting the difficulty of propagating sparse rewards across the state space.

Touches per episode increased from near zero early in the beginning of training to approx. 0.3–0.5 touches per episode by the end of training, measured with a 20 ep moving average. This improvement demonstrates that the agent learned to contact the ball more frequently than random exploration would allow.

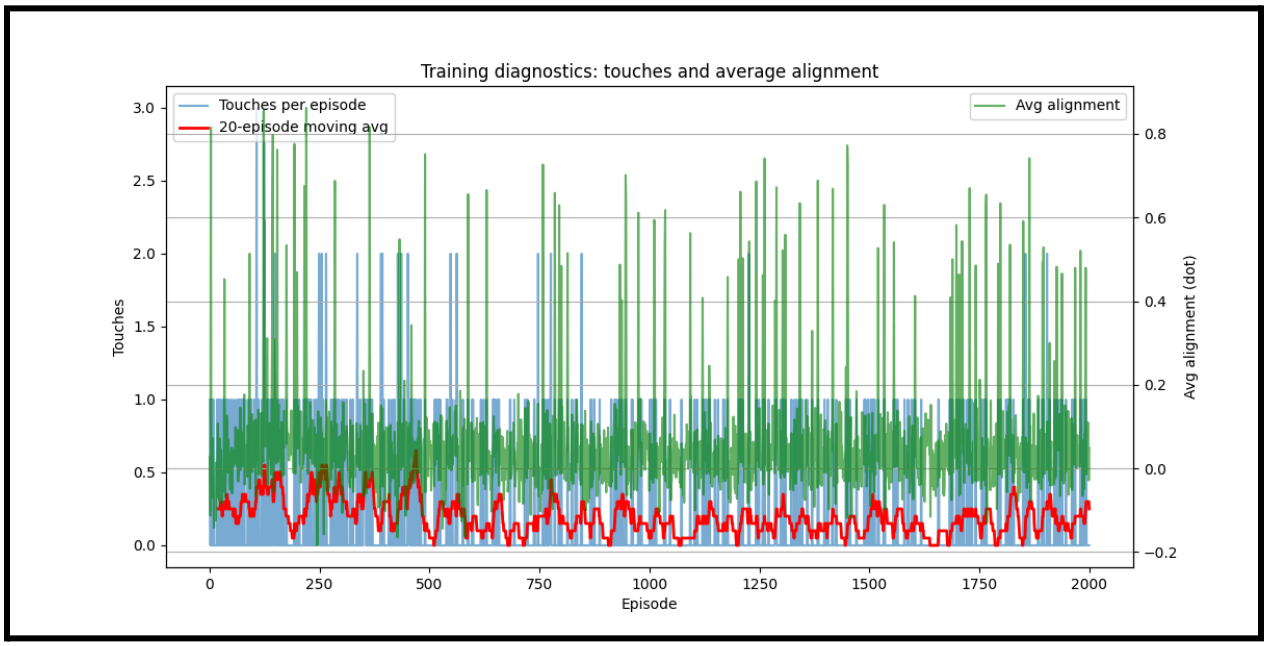

Figure 3.3: Rise from 0 to 0.3–0.5 indicates improved ball-seeking behavior. Average alignment increases, consistent with steering rewards guiding orientation toward the ball.

3.3 Interpretation

These qualitative and quantitative results together underscore the significance of the capability of the table-based Q-learning approach incorporating discretization techniques and reward shaping in terms of positive behavioral changes occurring in the game of Rocket League. The agent learned to ground the ball effectively, achieving a higher number of touches. The action values of Q concentrated in states in which the goal of touching the ball was possible. Unfortunately, there was variability in values of maximum Q-learning values.

4. Discussion

The experiments demonstrate that a tabular Q-learning agent, even with a highly reduced state and action space, can learn task-relevant behaviors in Rocket League. Across multiple runs, the agent consistently discovered value in states close to the ball and achieved intermittent touches. This confirms that the Bellman update rule, combined with ε-greedy exploration and reward shaping, can propagate sparse signals backward through the discretized state space. Maximum Q-values varied widely across runs, ranging from 6.5 to nearly 10, and touches per episode plateaued at relatively low levels. These results highlight the fragility of tabular Q-learning in continuous, partially observable domains.

4.1 Limitations

Several limitations explain these outcomes. The sparse reward structure makes credit assignment difficult. Watkins and Dayan's convergence theorem requires repeated sampling of all state-action pairs under bounded rewards (1992, pg. 282–285). In Rocket League, touches and goals are rare, meaning many upstream states are never revisited often enough to propagate value reliably.

Discretization introduces aliasing. By representing the state with only distance and speed, we can see critical information such as orientation, lateral offset, and boost status. As the continuous state Q-learning paper notes, uniform discretization can cause "reachable mismatch" (Alaoui & Saoud, 2024, Section 3) where trajectories cross cell boundaries and deviate from the idealized representation of cell centers. This mismatch undermines the Markov property and produces non-stationary returns within a single bucket.

Exploration coverage was incomplete. Runs typically visited only 30–45% of the discretized grid. Sutton and Barto emphasize that Q-learning is off-policy and converges to the optimal policy only if sufficient exploration occurs (2018, Ch. 6.5). In practice, ε-greedy exploration decayed too quickly, leaving many state-action pairs undersampled. Finally, learning exhibited high variance across seeds. Differences in spawn positions and stochastic exploration trajectories produced substantial variability in outcomes. This sensitivity reflects the instability of tabular methods in high-dimensional domains.
Implications

Despite these limitations, the agent did demonstrate learning. Concentrated Q-values near the ball and upward trends in touches per episode show that even simple tabular methods can extract meaningful behavior when combined with reward shaping. This makes tabular Q-learning a valuable pedagogical baseline: it reveals how discretization, shaping, and exploration interact, and it surfaces the structural challenges that must be addressed by more advanced methods.

4.2 Directions for Future Work

Several concrete next steps emerge from this analysis. In the short term, replicate experiments with multiple seeds should be run to quantify robustness. Adjusting ε-decay schedules or adding novelty bonuses could improve exploration coverage. Extending the discretizer to include coarse orientation or alignment bins would reduce state aliasing without exploding the grid size.
In the medium term, experience replay could stabilize updates and improve sample efficiency. Function approximation methods such as Deep Q-Networks (Mnih et al., 2015) or policy

gradient algorithms like Proximal Policy Optimization (Schulman et al., 2017) should be explored to handle continuous observations and actions directly.

5. References

Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. Machine Learning, 8(3–4), 279–292. https://doi.org/10.1007/BF00992698

Sutton, R. S., & Barto, A. G. (1998). Reinforcement learning an introduction. MIT Press.

Alaoui, S. B., & Saoud, A. (2024). How to discretize continuous state-action spaces in Q-learning: A symbolic control approach. arXiv.Org.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. Nature (London), 518(7540), 529–533. https://doi.org/10.1038/nature14236

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. arXiv.Org

6. GitHub Link

https://github.com/2025F-COMP3106/project-group-122