# Achieving Human-Level Results in a Physics-Based Game Environment Using Reinforcement Learning in Rocket League

Harry Xie

***Abstract*—The abstract goes here.**

## I. Introduction

ARTIFICIAL intelligence (AI) has wide applications, ranging from medical diagnosis to robot-control in the production of high-precision parts. The widespread use of AI use falls under "weak AI", where the AI is specialised for a specific task. This enables achieving super-human performance at these tasks.

Reinforcement learning (RL) is an approach to machine learning (a subset of AI) where an agent explores an environment, to learn which actions to take to maximise its received reward [1]. RL has achieved ground-breaking results in AI, such as in Atari games [2], Go [3], StarCraft II [4], and Dota 2 [5]. Not only applicable to video games, RL is also increasingly being applied to real-world physical applications such as self-driving cars [6], and robotic hand manipulation [7].

Inspired by the application of RL to specific sub-tasks within an environment, this work aims to investigate the use of RL in novel problems within a physics=based environment. While there exist frameworks that support this using simple toy problems [8], Rocket League[1] offers the opportunity to investigate an environment with competitive human players.

This investigation aimed to maximise the success of the agent, and does not serve as a systematic benchmarking of RL.

Section II provides a brief description of RL, and the two algorithms used in this investigation. Section III then details the RLBot framework used to interface with the game, and the limitations that result. Finally, section IV contains the results achieved by RL in various tasks attempted.

## II. Reinforcement Learning

In RL, an agent interacts with an environment by taking an action $a$ at every step, given an observation $o$ at that step. The observation, is a function of the environment's state $s$, $o = o(s)$, and may be a partial representation of the state of the environment. The initial state can be fixed or generated randomly. That is, $s_0 \sim \rho_0(\cdot)$, where $\rho_0$ is the start-state distribution [9].

While Rocket League's physics ticks are deterministic, and can ideally be formalised as $s_{t+1} = f(s_t, a_t)$, the framework's limitations as described in Sec. III results in $s_{t+1}$ becoming a function of additional factors such as step-duration and the synchronisation between steps and physics ticks. This causes an increase in variance, and the formalism in effect becomes a probabilistic one; $s_{t+1} \sim P(\cdot|s_t, a_t)$.

Furthermore, while $s_{t+1}$ in Markov Decision Processes are a function of the most recent state and action only, Rocket League has an additional quirk where a single action where "boost" is active results in four physics ticks where boost is active.

### A. Policy

The method with which the agent decides what action to take is called a policy. Policies act as a mapping from states to actions, and are denoted with $\pi$. The actions taken by an agent with policy $\pi$ at state $s_i$ follow a distribution given by $a_i \sim \pi(\cdot|s_i)$ [9].

RL policies have to balance between exploration and exploitation. Policies have to take advantage of what they already know to obtain rewards (exploitation), but have to try out other actions to learn about possibly-better actions. While there exist several standard techniques that attempt to strike this balance, the exploration-exploitation dilemma remains unsolved, despite having been "intensively studied by mathematicians for many decades" [1].

### B. Algorithms

*1) Deep Deterministic Policy Gradient:* Deep Deterministic Policy Gradient (DDPG) is an off-policy actor-critic algorithm that handles continuous action spaces. The algorithm consists of making use of two models; an "actor" model that tries to suggest the best action at each step, and a "critic" model that estimates the maximum total discounted future reward achievable with that action (the Q-value).

*2) Twin Delayed DDPG:* Twin Delayed DDPG (TD3) [10] is an iteration of DDPG that aims at being more robust in convergence. As DDPG often fails by overestimating Q-values, TD3 includes a couple of workarounds preventing this.

1) Clipped Double-Q Learning. Two independent models are trained in parallel to estimate the true Q-value. The target Q-values used in training are calculated as the minimum Q-value between the two trained models.
2) Delayed Policy Updates. The policy network is updated once for every two updates of the critic network, resulting in policy network updates using value estimates that are of lower variance.

[1]In this paper, the RL abbrieviation is used exclusively to refer to reinforcement learning to avoid confusion. Rocket League will be spelled out in full, or referred to as "the game".

3) Target Policy Smoothing. Noise is added to the target actions during training to prevent the policy model from over-fitting to erroneous narrow peaks in the value estimates.

Ablation studies showed that clipped double-Q learning provided the most significant improvement, while the latter two additions provided more minor, but non-negligible improvements [10].

## III. RLBot

### A. Framework

The RLBot framework allowed for the RL agent to interact with the game [11].[2] The framework prompts the agent for an action at a fixed tick-rate (usually 120 Hz or 60 Hz), and provides a packet of game information whenever it does so.

While the framework enables speeding up the game, speeding up the game past $4\times$ was known to cause visible physics issues.

### B. Existing Bots

There exist many hardcoded (non-RL) bots created by the RLBot community. These focus mostly on 1v1 play, and can achieve good performance in specific tasks such as dribbling. The use of RL has been previously experimented with in the much more complex problem of a 1v1 game, though this achieved little success.

## IV. Applications

### A. Shooting: Challenger

Implemented as an in-game training pack, Linkuru's challenge consisted of completing 33 consecutive shots [12]. These shots ranged from trivial pathing to the ball and getting a ball hit, to more complicated shots requiring well-timed flips or highly-optimised pathing. In many of these shots, the first touch was the most important and had the tightest margins. After a good first touch, the ball could then be caught and carried to the goal in a dribble; the post-first-touch dribble is a task likely already solved by existing hardcoded bots). Optimising this first touch using RL would then help reduce the custom code written for each shot.

To provide a denser reward to the agent, "ghosts" were created; controls and car locations were recorded during a successful attempt. The distance between the recorded car and the agent's car during an episode was added as an additional reward. Additionally, episodes were probabilistically ghost-controlled. A ghost-controlled episode reproduced the recorded controls for the ghost, although this does not identically reproduce the initial recorded episode due to inaccurate frame timings. The probability that an episode was ghost-controlled was set manually based on how well the agent was doing; the more guidance the agent needed, the more likely it was that episodes are ghost-controlled.

[2]The RLBot framework is supported by Psyonix, the developers of Rocket League. While this work utilised an older version of the framework that used DLL injection to interface with the game, the framework has since been migrated to use an in-game Psyonix-developed API.


Fig. 1. Snapshot of Challenger's training on Shot 1. The ghost's location can be seen as the white square between the car and the ball. This is a snapshot from episode 8805 linked in Tab. I.

*1) Shot 1: Simple Shot:* This shot involved timing a drive-up to hit the ball, which was slowly flying across the face of the goal.

Figure 1 shows a training episode of this shot. The distance between the car's current location and the ghost's location (as indicated by the white square) was a factor in the calculation of the reward.

The agent showed signs of learnt behaviour within hundreds of episodes, such as driving and flipping towards the ball. However, it only managed to score a handful of times in 1000 episodes, with some seeming coincidental. However, given the signs of learnt behaviour, it was assumed that the shot could be consistently scored given its loose margins (due to the slow ball and proximity to the goal), if training were continued for 50000 episodes, and the best-performing model was chosen.

For this shot, roughly 10 human-controlled ghosts were recorded. However, these ghosts differed significantly in pathing, and the differences (and low count) might have hindered learning.

*2) Shot 33: Ceiling Flip-Reset:* After the small successes on Shot 1 of the challenge, the more lofty goal of Shot 31 was attempted. This shot included a requirement of a ceiling-reset (which involves flying to the ceiling and making contact with the ceiling with all four wheels) before scoring the goal. This is shown in Fig. 2.

Here, my abilities were not sufficient to complete the shot with the flip-reset requirement. However, a hardcoded bot by RLBot developer Kipje13 managed to complete the shot, flip-reset included, with consistency. He kindly recorded his controls and sent them over as ghost controls. These ghosts however deviated slightly as the controls took control of the car before the car settled[3]. This meant that while the ghost-controlled episodes managed to hit the flip-reset (though not consistently), they always fell short of hitting the ball[4]. This can be seen in Fig. 3.

For this shot, the game was set to 120 Hz in an attempt to better follow the ghost - which was recorded at 120 Hz. How-

[3]This is opposed to Challenger, whose implementation included a second of wait time to allow the car to settle. The challenge did not include any specifications on this point, and it was of little significance to human-play, where exact controls are not reproduced, and minor adjustments are invariably made

[4]While the ghost-controlled episodes fell short of hitting the ball, the dense-reward-adjustments were done based on the positions of the recorded ghosts; that is, reward was maximised when the agent follows the path of the car that hit the ball.


(a) Starting Position (b) Ceiling Reset (c) Ball Hit

Fig. 2. Snapshots of Challenger's training on Shot 33. The ghost's location can be seen as the white square in the starting position, but is not visible subsequently as it is behind the car. This is a snapshot from episode 9563 linked in Tab. I.


Fig. 3. Snapshot of a ghost-override episode during Challenger's training on Shot 33. The ghost's location can be seen as the white square in the top left, with the car's location deviating significantly despite the replication of controls from the ghost during this episode. This is a snapshot from episode 12555 linked in Tab. II.

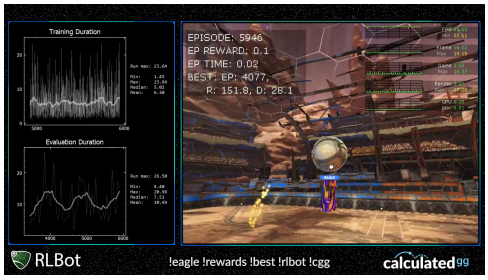



Fig. 4. Snapshot of Eagle's training showing the starting position for the episode. This is a snapshot from episode 5946 linked in Tab. IV.

ever, performance issues were visible and unstable framerates impacted performances in some episodes (see episode 9563 in Tab. II).

### B. Air-dribbling: Eagle

A 3-dimensional, more complex analogue to the typical CartPole problem, the aim of the agent in this task was to keep the ball in the air through the use of rotations and boost. Through pitch, yaw, and roll controls, the car could be pointed in a suitable direction, and boost was used to keep the car and the ball in the air.

The starting state for this task was randomly generated. While the car and ball's position were fixed, the car's yaw and roll was uniformly sampled over $0$ to $2\pi$, while its pitch was uniformly sampled over $\frac{\pi}{2} \pm 0.1\pi$. This is shown in Fig. 4.

Reward was defined similarly to CartPole, with $+0.1$ every step the agent survives, and $-1$ on the frame where the episode ends. An episode ended when the car-ball distance exceeded a fixed threshold, or when the ball dropped below a separate height threshold. Additional limits for the horizontal $x$ and $y$-coordinates of the ball were introduced after undesirable behaviour (visible in Fig. 5) resulted in technical issues where many non-existent episodes were detected as the game failed to set the state during a goal replay.

As a benchmark for human-level achievements, the human achievements during the same year as this investigation include the following: LEDxRL3 first broke the minute-mark in January 2019 [13], and then surpassed that with another record of $237\,\mathrm{s}$ [14] two months later. YouTuber dl g then broke this record with a $5127\,\mathrm{s}$ (roughly $85\,\mathrm{min}$) air-dribble in July [15].

## V. Discussion

### A. Technical Limitations

*1) Game Interaction:* While most RL environments are agent-controlled, this passive agent setup meant that frames could be dropped, leading to fluctuating step durations. The game also runs at an internal $120\,\mathrm{Hz}$ physics tickrate, and no work was done to ensure any synchronisations.

In practice, the game was set to run at a maximum of $120\,\mathrm{Hz}$ or $60\,\mathrm{Hz}$, and there were frame drops every couple of seconds due to limited computational capabilities. This can be seen in the training clips in Appendix B.

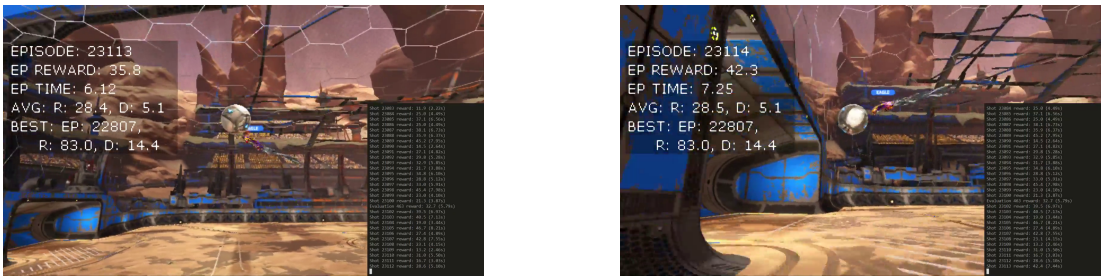
Additionally, the game was not sped-up during training as the loss in viewability was deemed more significant than the increase in data collection. Additionally, any game speed-ups would increase computational requirements, and this would have likely resulted in extremely unstable step-durations.

Overall, attempts were made to maximise training batch size while sustaining reasonably stable framerates.

A lock-step function has recently been introduced to the RLBot framework. This could provide a solution to the issues stemming from insufficient performance described above, and is a direction for further efforts.

*2) Game Restarts:* The game had to be restarted about every 50 hours. Models were saved regularly, and training was generally restarted using the saved models. However, the decision was sometimes made to restart training from scratch either due to new inputs or due to the guess that a model had reached a local maximum. The latter often took the form of the action becoming observation-independent.

While these game-restarts likely slowed training due to the loss of the replay buffer and the actor optimiser state, the




(a) Episode 23113 (b) Episode 23114

Fig. 5. Snapshot of Eagle's training showing significant horizontal movement. These episodes display undesirable behaviour, and instigated the introduction of the additional failure modes of horizontal $x$ and $y$-limits. Links to these episodes can be found in Tab. III. a) While the car has visibly lost control of the ball by this point, it managed to prolong the episode by falling after the ball, achieving a duration of $7.44\,\mathrm{s}$. b) The agent again visibly loses control of the ball, scoring the ball this episode. This introduced technical issues.

former was not a significant loss as the replay buffer could simply be rebuilt by re-running a warmup. Additionally, the loss of the actor's optimiser state is suspected to effectively serve as a cyclical learning rate, with the trained models being warm-restarted.

### B. Learning Algorithm

While DDPG achieved a maximum performance of roughly sub-$30\,\mathrm{s}$ episodes and training episode averages topped out at $7\,\mathrm{s}$, TD3 achieved a maximum episode length roughly $50\times$ higher with evaluation episode averages of roughly 2 minutes. This is in spite of the fact that the TD3 agent was working in stricter conditions (with extra failure modes limiting the area it could go), while the DDPG agent had episodes where it significantly increased its episode length despite having visibly lost control of the ball by falling in the same direction. Additionally, the DDPG agent often had to be restarted from scratch (i.e. using new model weights) to improve performance, with catastrophic forgetting happening regularly. Significantly less effort was spent doing the same for the TD3 agent, which was more robust in converging than the DDPG agent.

The improvements by the TD3 agent resulted in greatly increased control of the ball, with limited ball movements during episodes. This can be seen in Fig. 6.

### C. Air-dribbling Technique

Whilst humans aim to keep the ball touching the ceiling and do so for the vast majority of their air-dribbles [13]–[15] (dl g for example only visibly leaves the ceiling in the last $2\,\mathrm{min}$ of his $85\,\mathrm{min}$ air-dribble), the TD3 agent never reaches that height. This is likely a result of the inherent pessimism of the algorithm, where the target Q-values are taken as the minimum of the two calculated. With similar exploration parameters, the TD3 agent is more unlikely to learn to do the ceiling pinch when compared to the DDPG agent, but any long-term ill effects limiting learning are only speculative. The TD3 agent has effectively managed to solve the task, with many evaluation episodes lasting minutes while each ball hit has an effect timescale of about a second. The eventual lost of ball-control is possibly due to game lags.

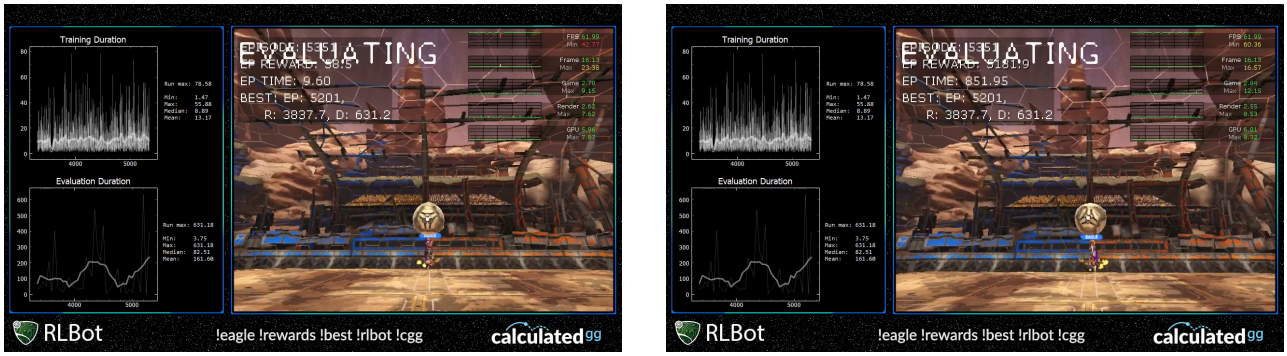
Preliminary tests involving shifting the starting positions for the ball and car closer to the ceiling have not yielded any significant results in learning the ceiling-pinch technique.

## VI. Conclusion and Outlook

This work explored the use of RL at various specific tasks in Rocket League. Some success was achieved in completing (though this was inconsistent and does not represent solving of the task) two shots of Linkuru's challenge. Much greater success was achieved in the use of TD3 at the task of air-dribbling, with the trained agent being competitive with human world-records.

Further training can be done at these tasks and will likely yield improved results. The application of RL to other tasks in Rocket League such as pathfinding will be more applicable to competitive play, unlike the tasks investigated in this work, which are largely academic.

This work also utilised an older version of the RLBot framework, it is strongly recommended that follow-up works use later versions, as new features have been introduced. The lock-step feature in particular is likely worth implementing.




(a) 9.6 s (b) 852.0 s

Fig. 6. Snapshot of Eagle's evaluation showing extremely limited ball movement. A link to the clip of this episode (episode 7651) can be found in Tab. IV.

## APPENDIX A
## SETUP OF TASKS

Normalisation was done for the individual inputs, roughly based on their maximum possible values. For example, the car's $x$-coordinate was divided by 4000, its $y$-coordinate by 6000, and its $z$-coordinate by 400.

Care was also taken to scale the rewards such that the main objective provided an extremely large reward in comparison to the rewards provided for additional guidance (such as the distance from ghosts).

### A. Challenger Shot 1

*1) Input:*

- Car location
- Car velocity
- Car rotation
- Car angular velocity
- Car jumped status
- Ball location
- Ball velocity
- Time elapsed in episode

*2) Reward:*

- Goal: $+50$; End without goal: $-50$
- Distance from ghost$/10000$
- *Boost: $-0.04$
- *Jump: $-5$
- *First touch: 20

Asterisk indicates reward was added at a later time.

### B. Challenger Shot 33

*1) Input:*

- Car location
- Car velocity
- Car rotation as quaternions
- Car angular velocity
- Car jumped status
- Ball location
- Ball velocity
- Time elapsed in episode

*2) Reward:*

- Goal: $+50$; End without goal: $-50$
- Distance from ghost$/10000$
- First ceiling reset: 20
- First touch, if ceiling reset has been performed: 40
- Jump: $-5$

### C. Eagle

*1) Input:*

- Boost, Pitch, Yaw, Roll at previous physics frame
- Ball's displacement from the car
- Boost action from 3 previous steps
- Car location
- Car velocity
- *Car rotation as sine and cosine for each Euler angle
- Car angular velocity
- Car jumped status
- Ball location
- Ball velocity
- Time elapsed in episode

Asterisk indicates that this was changed (at an unknown time before 08 May 2019), from a representation with quaternions.

*2) Reward:*

- Not done: $+0.1$
- Done: $-1$

The episode is done when the ball falls such that its $z < 500$, or if the car-ball distance was greater than 400. The additional failure modes where the ball's $x > 2000$ or $y > 2000$ were added to prevent the issue dealing with the goal-scoring issue seen in Episode 23114 (link available in the second column of Tab. III).

## APPENDIX B
## TWITCH CLIPS

These clips are provided for reference, and as a whole are indicative of the training process. Note that training parameters including but not limited to policy noise, learning rate, and reward parametrisation varied throughout the training.

The episode numbers do not represent the total number of training episodes experienced by the agent. The game had to be restarted about every 50 hours, and the episode count was reset whenever this occurred.

### A. Challenger

*1) Shot 1:*

*2) Shot 33:*

### B. Eagle

*1) DDPG:*

TABLE I
TWITCH CLIPS FROM CHALLENGER'S TRAINING ON SHOT 1

| Episode(s) | Event(s) | Clip | Notes |
|---|---|---|---|
| 84 | Ball hit | Link | Possibly coincidental, flips towards ball |
| 268 | Goal | Link | Flips towards ball, backwards goal |
| 412-417 | Training | Link | Demonstration of ghost-override episodes |
| 734 | Goal | Link | Drives towards ball, jumps |
| 820 | Goal | Link | Flips forwards, reverses slightly to prevent overshooting, flips to score |
| 39 | Goal | Link | Likely coincidental, backwards flip to score |
| 4925 | Goal | Link | Controlled movements, flips to score |
| 6008 | Goal | Link | Aerial goal |
| 6264 | Goal | Link | Correction after sideflip, flips to score |
| 7084 | Goal | Link | Aerial goal, includes 2 episodes of training misses |
| 7459-7469 | 6 Goals | Link | 7460, 7462, 7463 (GO), 7465, 7466, 7469 |
| 7485-7489 | Goal | Link | 7488, clean drive |
| 8801-8806 | 3 Goals | Link | 8802, 8803, 8806 |
| 12230-12233 | 2 Goals | Link | 12231, 12232 |

TABLE II
TWITCH CLIPS FROM CHALLENGER'S TRAINING ON SHOT 33

| Episode(s) | Event(s) | Clip | Notes |
|---|---|---|---|
| 3347-3348 | Pathing | Link | Tracks closely to ghost |
| 3353 | Ceiling Reset | Link | Touches wheels on the ceiling |
| 4760-4763 | Ball Hit | Link | 4761. Does not get ceiling reset on this episode. End of 4760 (GO) is indicative of GO episodes. |
| 4730-4734 | Ball Hit | Link | 4732. Gets a ceiling reset on this shot, and on episode 4734. |
| 8408-8409 | Ball Hit | Link | 8409. Touches the ceiling, but does not get a reset. Ball almost scored. |
| 9670-9671 | Ceiling Resets | Link | The difficulty of getting clean ceiling resets can be seen in 9670, where the car loses significant momentum in the process. |
| 9701-9703 | Ceiling Reset | Link | 9701. An evaluation run, close to hitting the ball. |
| 10301 | Ball Hit | Link | An evaluation run, might not have gotten a ceiling reset. |
| 16517 | Ball Hit | Link | Did not get a ceiling reset, shot is too high, ball ends on top of the roof of the goal. |
| 16833-16834 | Ball Hit | Link | 16834. Got a ceiling reset, shot fell slightly short of goal and was slightly wide. |
| 19343 | Ball Hit | Link | Did not get a ceiling reset |
| 3867 | Goal | Link | Got the ceiling reset, clean goal. |
| 12555-12557 | Goal | Link | 12557. Did not get the ceiling reset. 12555 ghost override behaviour can be seen, along with in-game FPS indicator. |
| 5125 | Goal | Link | Got the ceiling reset, clean goal. |
| 9563 | Goal | Link | Got the ceiling reset, clean goal. Extremely unstable performance visible in FPS indicator |
| 11718 | Ball Hit | Link | Got the ceiling reset, clean shot falling just short of the goal. Possible sign of correcting path toward the ball after the ceiling reset. |

TABLE III
TWITCH CLIPS FROM EAGLE'S TRAINING WITH DDPG

| Episode(s) | Duration | Clip | Notes |
|---|---|---|---|
| 6712-6735 | NIL | Link | Unstable 120 Hz. Indicative of untrained behaviour. |
| 2503-2506 | $\sim 2$ | Link | 2503. Slightly trained behaviour of getting first hit, using boost. |
| 11383-11395 | NIL | Link | Multiple ball hits. |
| 13268-13281 | NIL | Link | |
| 15697-15708 | NIL | Link | |
| 18424-18425 | $\sim 3$ | Link | 18425 |
| 18842-18854 | NIL | Link | |
| 18863-18873 | NIL | Link | |
| 21829 | $\sim 4$ | Link | |
| 22389-22397 | NIL | Link | |
| 154-162 | NIL | Link | |
| 252 | $\sim 3$ | Link | Unstable 120 Hz |
| 8382-8392 | NIL | Link | |
| 9765-9774 | $\sim 4$ | Link | 9773 |
| 9768-9778 | | Link | |
| 9768-9787 | | Link | |
| 9771-9780 | | Link | |
| 12807-12821 | NIL | Link | |
| 12877-12878 | $\sim 5$ | Link | |
| 935-943 | 3.7 | Link | 940. Intrroduced overlay showing duration. |
| 15125-15130 | 5.1 | Link | 15130 |
| 1885-1886 | 4.8 | Link | Stable 30 Hz |
| 3199-3201 | 4.9 | Link | 3200 |
| 166-186 | NIL | Link | Unstable 120 Hz due to training. |
| 20729-20734 | 5.5 | Link | 20731. Somewhat stable 60 Hz |
| 21284-21288 | 5.6 | Link | 21285 |
| 22055-22058 | 5.7 | Link | 22058. Not fully visible due to camera angle. |
| 22850-22852 | 6.4 | Link | 22851 (Evaluation) |
| 28139-28141 | 6.7 | Link | 28140 |
| 29962-29964 | 6.9 ;) | Link | 29963 |
| 32236-32238 | 7.1 | Link | 32237 |
| 32873 | 7.7 | Link | |
| 33183 | 8.5 | Link | |
| 39249-39250 | 8.6 | Link | 39249. Average duration over 4s. |
| 39662-39663 | 8.7 | Link | 39662 |
| 39691 | 14.8 | Link | Not fully visible due to camera angle. |
| 26 | 9.8 | Link | |
| 949 | 8.5 | Link | Unstable 60 Hz due to training. Average duration of 4.5s. |
| 1294-1295 | 9.2 | Link | |
| 2749-2753 | 8.0 | Link | 2751 (Evaluation). Extreme horizontal movement, close to goal. |
| 3383 | 12.5 | Link | Horizontal and vertical movement and correction. |
| 5201-5205 | 12.1 | Link | 5203. Precise control in stopping the ball during the dribble. |
| 5405-5406 | NIL | Link | |
| 5419-5423 | NIL | Link | |
| 5449-5455 | NIL | Link | |
| 5459-5464 | NIL | Link | |
| 5541 | NIL | Link | Frontflip |
| 5775-5776 | 12.8 | Link | |
| 11551 | 11.4 | Link | Evaluation |
| 12216 | 11.4 | Link | Significant horizontal movement, close to goal |
| 12499-12500 | 12.0 | Link | |
| 21558-21560 | 13.4 | Link | 21559 |
| 7992-7999 | NIL | Link | |
| 7999-8001 | 12.2 | Link | 8000 |
| 9581 | 14.0 | Link | |
| 9601-9606 | NIL | Link | |
| 17624 | 14.1 | Link | |
| 22807 | 14.4 | Link | |
| 23113-23114 | Goal | Link | 23114. Visible issue dealing with goals, many instantly-ending episodes. |
| 72346-72363 | NIL | Link | Bad performance |
| 6386 | 10.6 | Link | |
| 7102 | 10.7 | Link | |
| 7299 | 11.7 | Link | |
| 12085-12089 | NIL | Link | Average duration of 4.8s |
| 14437-14438 | 12.0 | Link | 14437 |
| 23015-23022 | NIL | Link | |
| 5031-5035 | NIL | Link | Average duration of 5.0s |
| 5205-5209 | NIL | Link | Average duration of 5.2s |
| 7910 | 12.4 | Link | Significant vertical movement, might have hit ceiling. |
| 13749-13750 | NIL | Link | 13750 contains a couple of controlled ball bounces. |
| 20716-20719 | NIL | Link | |
| 20768 | NIL | Link | |

| Episode(s) | Duration | Clip | Notes |
|---|---|---|---|
| 3975 | 12.2 | Link | |
| 10051 | 12.7 | Link | Evaluation |
| 20256-20260 | NIL | Link | Average duration of 5.8s |
| 19837 | 14.6 | Link | Significant direction changes and corrections. |
| 22389-22390 | 15.3 | Link | 22389 |
| 22474-22476 | NIL | Link | Average duration of 6.0s |
| 11301-11302 | NIL | Link | |
| 18091 | 16.4 | Link | |
| 1925-1928 | 15.5 | Link | 1925. Average duration of 7.0s |
| 4616 | 16.0 | Link | |
| 6901-6904 | NIL | Link | Consistent vertical movements across episodes. |
| 12849 | 16.0 | Link | |
| 19653 | 16.5 | Link | |
| 16624 | 17.1 | Link | |
| 7748 | 17.4 | Link | |
| 1245 | 17.7 | Link | |
| 1696 | 18.2 | Link | |
| 2329 | 19.2 | Link | |
| 3989 | 20.2 | Link | |
| 5426-5430 | NIL | Link | Average duration of 6.9s ;) |
| 5312 | 22.1 | Link | Touches the ceiling near the start |
| 11672-11676 | NIL | Link | Touches the ceiling for a couple of seconds in episode 11672. Gains significant height across episodes. |

TABLE IV
TWITCH CLIPS FROM EAGLE'S TRAINING WITH TD3

| Episode(s) | Duration | Clip |
|---|---|---|
| 2001 | 27.4 | Link |
| 2051 | 28.0 | Link |
| 5946 | 29.8 | Link |
| 7601 | 33.2 | Link |
| 7807 | 36.8 | Link |
| 51 | 108.7 | Link |
| 101 | 5:46 | Link |
| 3651 | 6:42 | Link |
| 4351 | 8:54 | Link |
| 5201 | 10:31 | Link |
| 5351 | 14:16 | Link |
| 7651 | 23:52 | Link |

*2) TD3:*

## ACKNOWLEDGMENT

I would like to thank the RLBot community, not only for the framework that enabled this work to be done, but also for their support in troubleshooting problems and providing ghosts. I would also like to thank Twitch chat for creating clips and for their kind words. Last but not least, thanks also go to Psyonix for the wonderful game, and for their continued support of the RLBot community.

## REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning An Introduction*. MIT Press, 2018.
[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, p. 529, Feb. 2015.
[3] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–503, 2016. [Online]. Available: http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html
[4] O. Vinyals, I. Babuschkin, J. Chung, M. Mathieu, M. Jaderberg, W. M. Czarnecki, A. Dudzik, A. Huang, P. Georgiev, R. Powell *et al.*, "AlphaStar: Mastering the real-time strategy game StarCraft II," *DeepMind Blog*, 2019.
[5] OpenAI, "OpenAI Five," 2018. [Online]. Available: https://blog.openai.com/openai-five/
[6] M. Bansal, A. Krizhevsky, and A. Ogale, "ChauffeurNet: Learning to drive by imitating the best and synthesizing the worst," *arXiv preprint arXiv:1812.03079*, 2018.
[7] M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray *et al.*, "Learning dexterous in-hand manipulation," *arXiv preprint arXiv:1808.00177*, 2018.
[8] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.
[9] J. Achiam, "Spinning up in deep RL," Nov. 2018. [Online]. Available: https://spinningup.openai.com/en/latest/index.html
[10] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," *arXiv preprint arXiv:1802.09477*, 2018.
[11] D. Sexton and RLBot Contributors, "RLBot: A framework that allows people to write their own Rocket League bots." [Online]. Available: http://www.rlbot.org/
[12] Linkuru, "FREE $3000 ALPHA ITEMS, if you beat this challenge..." Apr. 2019. [Online]. Available: https://www.youtube.com/watch?v=AMS01PdJ9Lw
[13] LEDxRL3, "World's first minute long air dribble in free play (67 seconds)," Jan. 2019. [Online]. Available: https://www.youtube.com/watch?v=VyDxq4cgupU
[14] ——, "237 second air dribble," Mar. 2019. [Online]. Available: https://www.youtube.com/watch?v=uxJUubAwywY
[15] dl g, "(5127:43 seconds) (85 minutes) air dribble world record," Jul. 2019. [Online]. Available: https://www.youtube.com/watch?v=BFMWr5PEuuc