DeepRacer: a Deep Dive in Deep Q-Learning

Quin Thompson
Department of Computer Science and Engineering
University of Washington
Seattle, WA
q5@uw.edu

Vasily Ilin
Department of Mathematics
University of Washington
Seattle, WA
vilin@uw.edu

Abstract

We present our exploration of training an autonomous race car in a simulated environment using deep Q-learning. We obtained a model that placed in the top 3% in the June competition and generalized to other tracks.

1 Motivation & Problem Definition

Autonomous driving is one of the biggest applications of Deep Q-Learning. DeepRacer [1] provides a framework to train a car in a simulated environment 1, compete in online races and download the trained model to a physical vehicle. We set out to place in the top of the virtual leaderboard [2] with a generalizable to other tracks model.

In DeepRacer participants have control over the hyperparameters 11, the reward function, the track 2 to train on, and the training time. Each training episode consists of the car driving on the virtual track until it goes off-track. The neural network topology is fixed at 3 hidden layers of unknown width. The console has detailed logs of each training episode and aggregated graphs of the reward (green), progress in training (blue) and progress in evaluation (red).

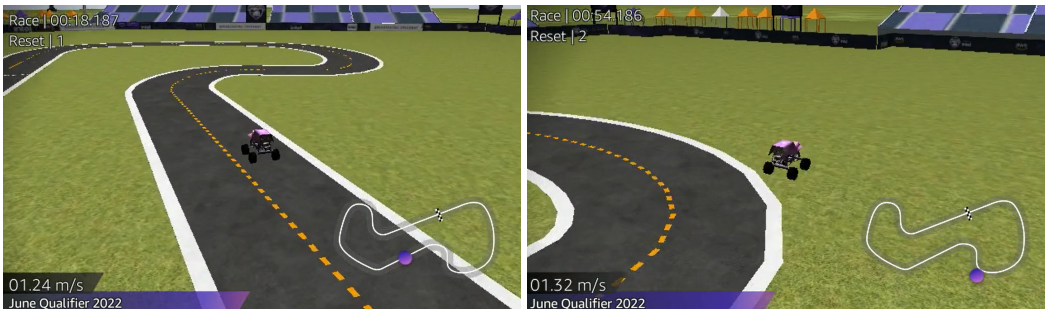

Figure 1: Normal progress (left) and crash (right)

2 Approach

We first tried the "feels good" approach of using the hyperparameters and reward function that seem good. This lead to unpredictable behavior 4: driving off-track as soon as possible to prevent negative reward, wild oscillations in performance, and driving fast without turning.

Systematic sweeps over hyperparameters and the reward function yielded better results. One of the latest models during sweeping was able to clear the top 5% of the leaderboard with only an hour of training. For comparison, some of the top-performing models train for 10+ hours.

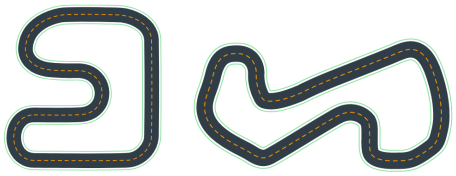

Figure 2: Ross Raceway (left) and BreadCentric Loop (right)

Figure 3: Legend

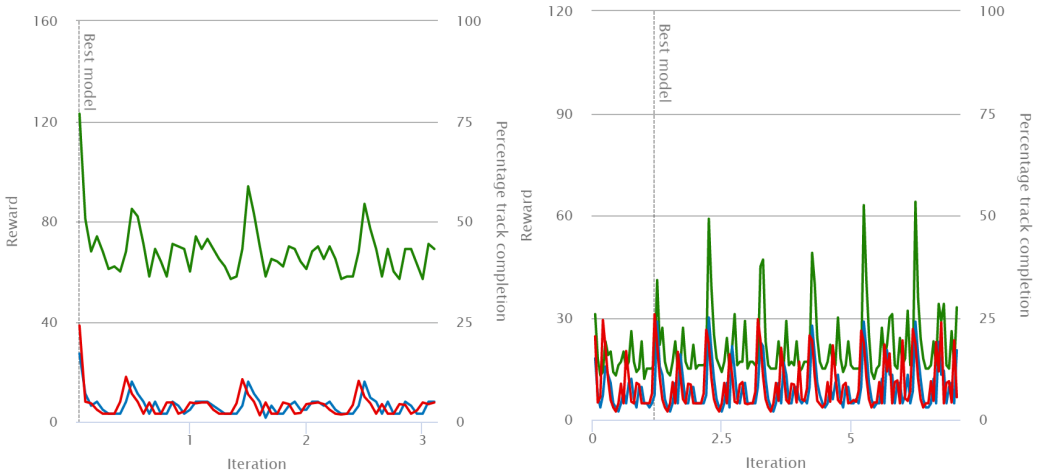

Figure 4: Results of badly chosen hyperparameters and reward function

3 Data

We sweeped over the following training parameters. See the appendix 8 for the full sweep data.

- Algorithm
- Action Space
- Max/Min Speed
- Reward Function
- Learning Rate
- Discount Factor
- Reward Function pt. 2
- Learning Rate pt. 2

See figures 5 and 6 for two particularly enlightening sweeps. The left graph shows that the best combination of minimum and maximum speed is (1,2). The right graph shows that the best reward function gives smooth reward for completing the track in fewer steps. Note the difference between the smooth and the non-smooth versions.

4 Results

At the time of writing, we are in top 3% (22nd out of 737) of the open Division leaderboard. Our model generalizes well, with top 15% performance on a completely different track. We have trained 83 models, ranging from 15 minutes to 10 hours of training, with an average of 1 hour.

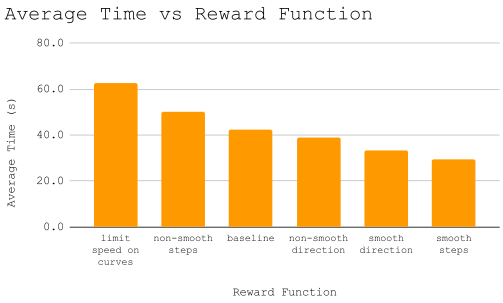

Figure 5: Reward sweep

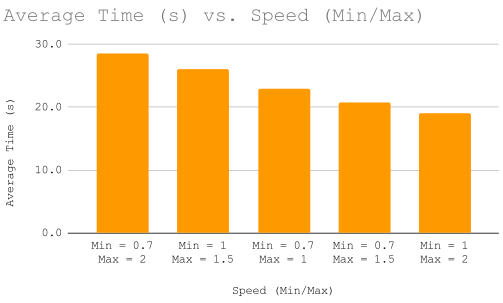

Figure 6: Speed sweep

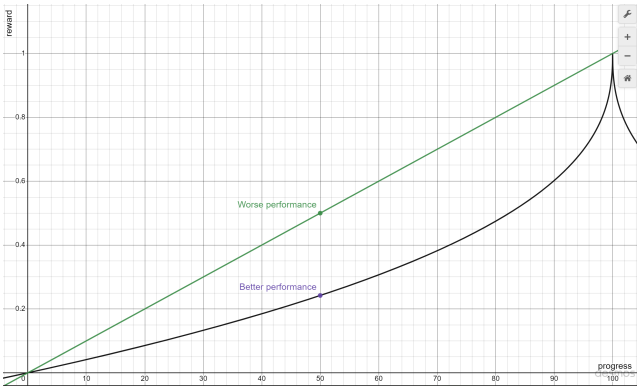

Figure 7: Gradient is key

5 Findings

5.1 Random vs Systematic

Systematic exploration of hyperparameters is more efficient than random trial and error. Because each meaningful training session takes at least an hour, and the number of hyperparameters is quite large, we did not do a grid search over the space of hyperparameters. Instead we sweeped over each hyperparameter one by one, using the defaults for the unexplored hyperparameters and the previously found values for the explored hyperparameters. This allowed us to make incremental improvements to out approach without losing the gains we made so far.

5.2 Reward Shaping

The reward function is the most important ingredient for a successful DeepRacer model. The following are our most important findings about the reward function.

1. Gradient is key. We saw the biggest improvements when we gave the reward function higher gradient in the critical region, while keeping it smooth. Figure 7 displays two approaches to reward progress on track: linearly and quasi-exponentially. The latter yielded much better performance. Intuitively, the higher the reward, the higher the gradient should be to incentivize the agent to move to the next state instead of just reaping high rewards at the current state.
2. Rewarding immediate behavior like speed, heading direction and centripetal acceleration does not work well. Figure 8 shows the effect of giving a (small) reward for slowing down before a curve. The agent ends up quickly learning to maximize the immediate reward and loses sight of the long-term goal of making progress on track and not crashing. However, rewarding staying near the central line of the track proved fruitful.

3. Reward must be non-negative most of the time. Otherwise the agent will maximize reward by going off-track to end the training episode. See figure 9 for a characteristic example.

A general tell-tale sign of a badly crafted reward function is non-correlation between the reward (green line) and the progress during training (blue line).

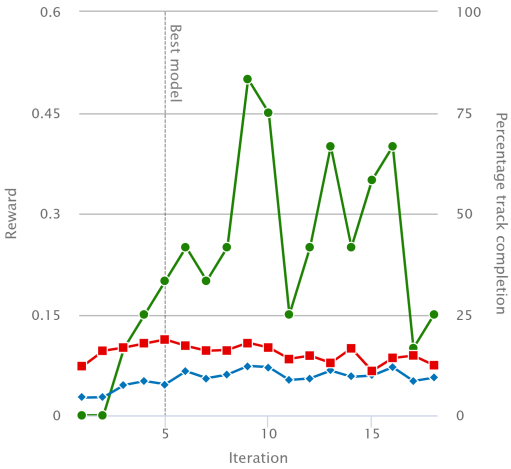

Figure 8: Overfitting for immediate rewards

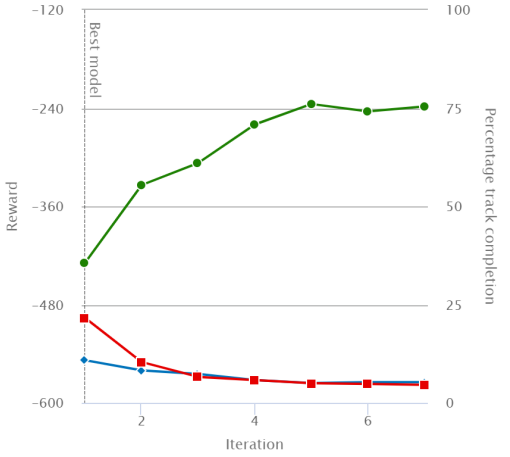

Figure 9: Negative rewards

5.3 Hyperparameters

The most important hyperparameters are learning rate and min/max speed. Higher learning rate works well for short training but causes divergence for longer (>1 hour) training. See figure 10 for an example of this. See 11 for the full list of hyperparameters.

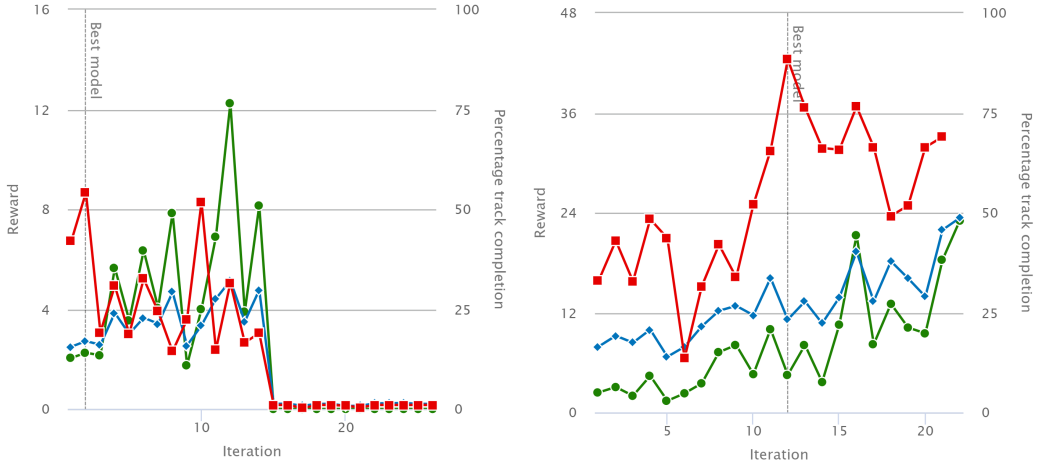

Figure 10: learning rate 0.001 (left) vs 0.00005 (right)

5.4 Action Space

It is important to limit action space as much as possible without sacrificing generalizability. With bigger action space convergence is either slow or does not happen. See 6 for an example of limiting the action space. Counter-intuitively, we saw better results with continuous action space despite it being a bigger action space. We conjecture that well-crafted discrete space would perform better due to faster convergence but will sacrifice generalizability to other tracks.

6 Broader Impact

The beneficiaries of this research are mainly Amazon, because we spent around $300 on the training and ourselves, because learned a lot about reinforcement learning through this experience. More broadly, autonomous driving chiefly benefits the individuals who can afford an expensive vehicle as well as the companies with enough R&D funds to afford early adoption of self-driving cars. Consumers may see slight price decrease over time as autonomous driving becomes cheaper and more reliable.

On other other hand, millions of professional drivers will lose their jobs. This is an unfortunate side-effect of automation and new technologies at large.

With self-driving becoming more prevalent, the ML engineers responsible for the training of virtual and physical models will need to pay more attention to biases in the data and algorithms. The failure to do so may result in self-driving agents that, in-effect, value the life of a white male over the life of a black female because of biased training data.

7 Acknowledgements

This work was funded by the CSE department of the University of Washington. We would like to thank Hannaneh Hajishirzi for the CSE 537 lectures.

References

[1] URL: https://aws.amazon.com/deepracer/.

[2] Aws-Deepracer-Community. *AWS-deepracer-community/deepracer-race-data: Historical data on AWS Deepracer League Virtual Races*. URL: https://github.com/aws-deepracer-community/deepracer-race-data.

[3] Ds. *How we broke into the top 1% of the AWS Deepracer Virtual Circuit*. Sept. 2020. URL: https://blog.gofynd.com/how-we-broke-into-the-top-1-of-the-aws-deepracer-virtual-circuit-573ba46c275.

[4] Daniel Gonzalez. *An advanced guide to AWS DeepRacer*. June 2020. URL: https://towardsdatascience.com/an-advanced-guide-to-aws-deepracer-2b462c37eea.

[5] *Writing great reward functions - bonsai*. Sept. 2017. URL: https://www.youtube.com/watch?v=0R3PnJEisqk.

8 Appendix

Figure 11 lists all hyperparameters available to us. Table 8 gives all our experiment data.

Hyperparameter	Advantage of higher values	Disadvantage of higher values	Default
Batch Size	More stable updates	Slower training	64
Epochs	Policy updates have higher effect	Slower training	10
Learning Rate	Faster training	May struggle to converge	0.0003
Entropy	More experimentation may lead to better results	May struggle to converge	0.01
Discount Factor	More stable model	Slower model	0.999
Episodes per Iteration	Improves model stability	Slower training	20

Figure 11: Hyperparameters

Algorithm	Trial 1	Trial 2	Trial 3	Average Time (s)
PPO	59.0	66.0	69.0	64.7
SAC	63.0	61.0	68.0	64.0
Action space	Trial 1	Trial 2	Trial 3	Average Time (s)
discrete	70.0	68.0	69.0	69.0
continuous	57.0	57.0	56.0	56.7
Speed (Min/Max)	Trial 1	Trial 2	Trial 3	Average Time (s)
Min = 0.7 Max = 2	23.8	26.2	35.6	28.5
Min = 1 Max = 1.5	27.1	29.7	21.6	26.1
Min = 0.7 Max = 1	25.1	24.9	18.7	22.9
Min = 0.7 Max = 1.5	20.6	22.6	19.3	20.8
Min = 1 Max = 2	16.7	21.3	18.9	19.0
Learning rate	Trial 1	Trial 2	Trial 3	Average Time (s)
0.0003	35.6	43.7	41.0	40.1
0.0005	39.1	39.4	38.5	39.0
0.001	33.7	33.4	43.0	36.7
Discount factor	Trial 1	Trial 2	Trial 3	Average Time (s)
0.9	43.3	48.5	41.1	44.3
0.99	43.7	33.9	39.8	39.1
0.995	41.8	52.6	46.7	47.0
0.999	33.7	33.4	43.0	36.7
0.9999	67.0	68.0	61.0	65.3
Reward function	Trial 1	Trial 2	Trial 3	Average Time (s)
limit speed on curves	56.0	70.0	62.0	62.7
non-smooth steps	49.0	56.0	46.0	50.3
baseline	34.8	47.0	46.0	42.6
non-smooth direction	45.0	33.0	39.0	39.0
smooth direction	33.0	34.0	33.0	33.3
smooth steps	27.0	31.0	31.0	29.7
Max Speed	Trial 1	Trial 2	Trial 3	Average Time (s)
2	44.0	33.0	35.0	37.3
2.5	37.0	52.0	49.0	46.0
3	42.0	54.0	55.0	50.3
3.5	66.0	61.0	61.0	62.7
10-hour Trains	Trial 1	Trial 2	Trial 3	Average Time (s)
MS 2, no direction, LR 0.001	33.0	33.0	38.0	34.7
SM 2, no direction, LR 0.001	35.0	37.0	30.0	34.0
MS 3, no direction, LR 0.001	35.0	31.0	29.0	31.7
MS 3, direction, LR 0.0005	36.0	28.0	28.0	30.7
Final sweep - Learning Rate	Trial 1	Trial 2	Trial 3	Average Time (s)
0.0003	29.0	31.0	30.0	30.0
0.0005	27.0	30.0	32.0	29.7
0.0001	29.0	29.0	28.0	28.7
0.00005	35.0	35.0	34.0	34.7
0.000005	28.0	28.0	31.0	29.0