Solving OpenAI's CarRacing environment using Deep Reinforcement Learning and Dropout

Patrik Gerber
University of Oxford
Corpus Christi College, Oxford, UK
patrik.gerber@ccc.ox.ac.uk

Jiajing Guan
George Mason University
Somewhere in Virginia?
jiajingguan@gmail.com

Elvis Nunez
Brown University
Somewhere in Los Angeles
elvis@brown.io

Kaman Phamdo
University of Maryland
Somewhere in Maryland
kaman@google.org

Nicholas Malaya
AMD Research
Somewhere in Austin
nicholas.malaya@amd.com

Tonmoy Monsoor
University of California Los Angeles
Somewhere in Los Angeles
mtonmoy@g.ucla.edu

Abstract

Deep Reinforcement Learning methods have seen many successes in recent years, ranging from solving classical video games to beating world class Go players. However, little progress has been made on the front of generalizability: successful models are trained for narrow, well-defined tasks, often using a vast amount of compute time. These models perform well in their specific task, but slight perturbations in the environment often cause disproportionate decrease in performance. Regularization methods have not yet been shown successful in tackling this issue of overfit. In this paper we attempt to give such a positive example, by applying the DDQN-algorithm with Dropout to solve OpenAI's CarRacing environment, using only a small subset of the state space for training.

1 Introduction

OpenAI Gym is a set of Reinforcement Learning testbeds including many classical video games and control problems. OpenAI Gym's CarRacing environment - from here on referred to as the 'car racing game' - is a simple, top-down view racing game, shown in Figure 1. The player has control of a race-car, and its goal is to visit the tiles making up the randomly generated track, as fast as possible. At each frame, the score changes by -0.1 if no tile was visited and by $\frac{1000}{N}$ otherwise, where N=Number of tiles on the track. The emulator returns 96 by 96 RGB screenshots of the screen at each frame. The game ends if the race-car visits all tiles on the track, or if the number of passed frames exceeds 1000. The environment is considered solved as per OpenAI's guidelines, if an average score of over 900 is achieved over 100 consecutive games. The car racing game was recently solved by [1], however to the best of our knowledge it is unsolved using RL methods.

In this paper we describe a Deep Reinforcement Learning algorithm that uses a convolutional architecture with dropout that successfully solves the game. Remarkably the model is trained on a limited environment made up of 3 tracks. This result shows that regularisation methods such as dropout can mitigate the overfit usually exhibited by these Deep Reinforcement Learning methods.

Preprint. Work in progress.

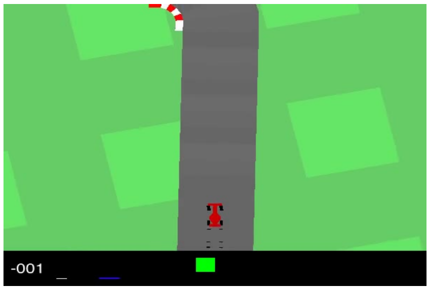

Figure 1: Screenshot of the car racing game.

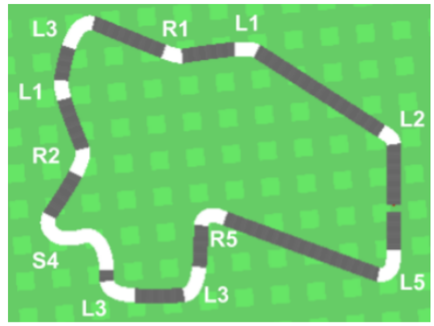

Figure 2: Characterization of different types of curves.

Our code can be found at *https://github.com/AMD-RIPS/RL-2018*, and video of the best performing model can be found under *TODO*

2 Method

2.1 Model

We use the DDQN-algorithm for our experiments, as first described in [4]. It is a simple extension of the celebrated DQN-algorithm, proposed in [2]. The architecture of the Q-network is that described in the original paper [2], consisting of 3 convolutional layers followed by 2 dense layers and the output layer. The input to the network is a $96 \times 96 \times 4$ image produced by stacking 4 consectuive frames of the game. Dropout (for a definition see [3]) is added to the second convolutional layer only, with a drop probability of 0.5.

2.2 Analyzing performance

We use two methods to analyze the performance of our models. The first one is to perform 100 consecutive test runs on randomly generated tracks, recording the average score. The second is to measure how the models perform on different types of curves in a racetrack. To do so, we developed a simple curve classification algorithm, which is demonstrated in Figure 2. Each curve is characterized as either a Left, Right, or S-shaped curve. An S-shaped curve is either a left turn followed by a right turn or a right turn followed by a left turn. Then, the steepness of the curve is ranked on a scale from 1 to 5, where 1 represents a very shallow curve and 5 represents a very steep curve. For each of these different types of curves we record the average percentage of tiles cleared.

3 Experiments

We train on three different environments: a single track, 3 different tracks and randomly generated tracks. We train on each of these environments both with and without dropout, giving a total of 6 training sessions. It is worth noting that the same fixed tracks were used for the first two environments when training with and without dropout. Training is over 3000 episodes and early stopping is used: only the best performing model is selected to be analyzed. Boxplots of the scores over 100 consecutive games for the 6 models are shown in Figure 3. The model trained on 3 tracks using dropout achieves an average score of over 906 with standard deviation $\sim$23, thereby solving the environment.

We used the curve characterization method to quantitatively measure whether applying dropout improved the ability to generalize to curves not seen during training. In particular, we analyzed the models that were trained on a single track, because these models only encountered a limited number of different types of curves during training. Even though both models were trained on the same track, the model with dropout performed better on each curve type. These results are demonstrated in Figure 4. This indicates that using dropout indeed allows for better generalization to curves not seen during training. Thus, dropout has the potential to be an effective regularizer in deep reinforcement learning problems.

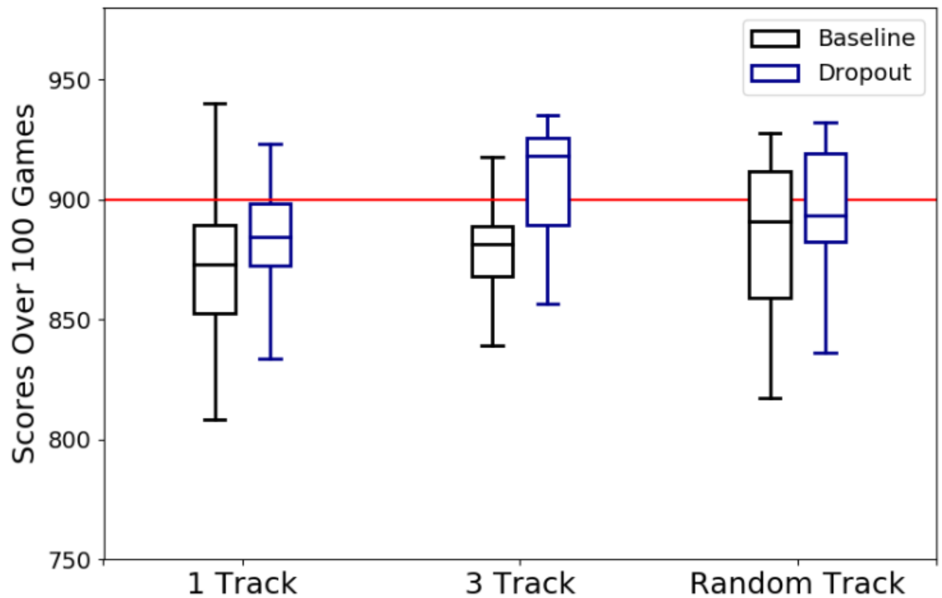

Figure 3: Comparison of the 6 models with 'Baseline' referring to models without dropout. The red line indicates the average score required to solve the environment.

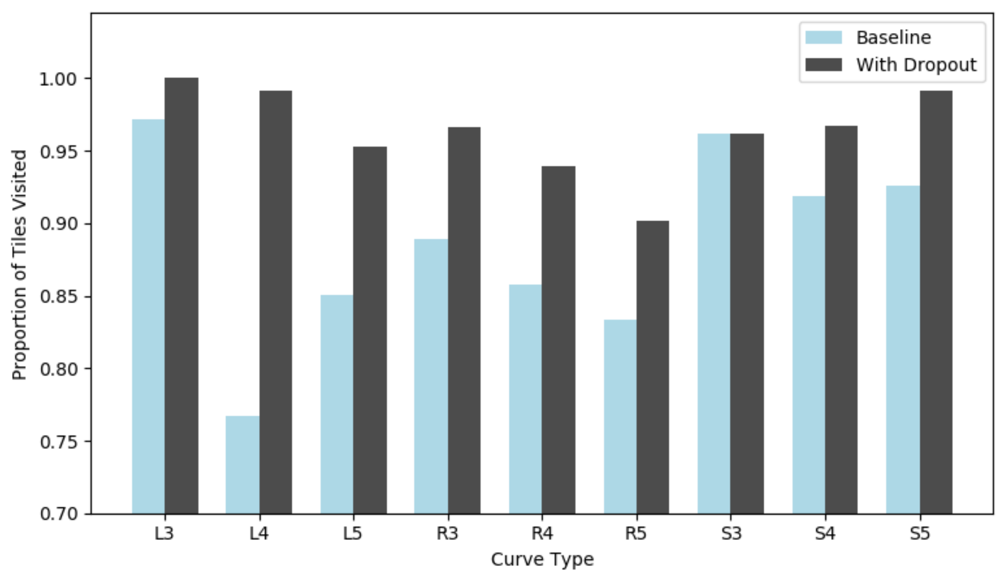

Figure 4: Comparison of performance of model trained on a single track, with 'Baseline' referring to no dropout.

4 Conclusion

References

[1] David Ha and Jürgen Schmidhuber. World models. *CoRR*, abs/1803.10122, 2018.

[2] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[3] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[4] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI*, volume 2, page 5. Phoenix, AZ, 2016.