Solving OpenAI's Car Racing Environment with Deep Reinforcement Learning and Dropout

Patrik Gerber*
University of Oxford
patrik.gerber@ccc.ox.ac.uk

Jiajing Guan*
George Mason University
jiajingguan@gmail.com

Elvis Nunez*
Brown University
elvis@brown.io

Kaman Phamdo*
University of Maryland
kaman@phamdo.com

Nicholas Malaya
AMD Research
nicholas.malaya@amd.com

Tonmoy Monsoor
UCLA
mtonmoy@g.ucla.edu

1 Introduction

Deep reinforcement learning methods have proven successful in solving well-defined computer games. In particular, the DDQN algorithm has achieved superhuman performance in several Atari games [4]. However, deep RL agents are limited by their sensitivity to perturbations in the environment. They can fail catastrophically when applied to environments that differ even slightly from where they were trained.

OpenAI Gym's CarRacing-v0 reinforcement learning testbed [3] is a is a top-down view car racing game where the goal is to visit all tiles of the racetrack as quickly as possible. The agent receives a reward of -0.1 at each time step and $\frac{1000}{N}$ for each track tile visited, where N is the total number of tiles in the racetrack. The state is represented by 96×96 RGB screenshots of the game. The game ends when the agent visits all tiles on the track or when 1000 frames have passed.

In this paper, we describe a deep RL algorithm that uses a convolutional architecture with dropout that successfully solves the game. Remarkably, the model is trained on a limited state space made up of 3 tracks. This result shows that regularization methods such as dropout can mitigate the overfit usually exhibited by deep RL. To the best of our knowledge, this solution is the first to solve the challenge using deep RL. The environment is considered solved by OpenAI's guidelines when the agent achieves an average score of 900 or above over 100 consecutive games. This task is particularly challenging because the racetrack is randomly generated for each game. The environment has recently been solved using a generative neural network model [1], but previous attempts using deep RL methods have not been successful [2].

2 Method

We adapted the DDQN algorithm [5] for our approach. The input to the Q-network is a $96 \times 96 \times 4$ tensor produced by stacking 4 consecutive frames of the game, and the output is five dimensional, corresponding to the elements of the discretized action space. As proposed in the original DQN algorithm [4], the architecture for the Q-network consists of 3 convolutional layers followed by 2 dense layers and an output for each of the five actions. Each hidden layer is followed by a rectified nonlinearity with dropout applied to the second convolutional layer only. We trained over a maximum of 3000 episodes corresponding to a training time of approximately 36 hours, and early stopping was used to ensure that the best performing model was selected for analysis.

The models were trained with varying observability of the environment. We performed experiments on three different subsets of the state space: a single (fixed) racetrack; three (fixed) racetracks; randomly generated racetracks (i.e. no restriction). The models were tested on the same 100 racetracks,

Preprint. Work in progress.

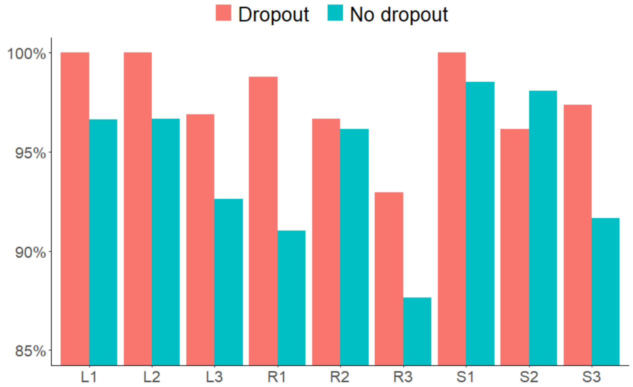

Figure 1: Proportion of tiles visited on different types of curves.

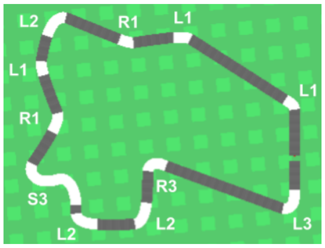

Figure 2: Different types of curves.

generated from random seeds 1 through 100. The results of the best performing agents from multiple training sessions corresponding to different combinations of dropout probability and state space can be seen in Table 1.

3 Performance Analysis

As shown in Table 1, the average scores over 100 random games improved in all three environments when dropout was applied to the Q-network. In order to investigate the generalizability of these models, we developed a curve classification algorithm. Each curve in the racetrack is charactertized as either a Left, Right, or S-shaped curve. Then, the steepness of the curve is ranked from 1 to 3, where 1 represents a shallow curve and 3 represents a steep curve. For each of these different types of curves, we record the percentage of tiles cleared.

Environment	**Dropout**	**Average Score**
1 Fixed Track	None	849.99 ± 78.72
3 Fixed Tracks	None	853.88 ± 127.71
Random Tracks	None	854.83 ± 107.14
1 Fixed Track	0.7	894.38 ± 24.5
3 Fixed Tracks	0.5	906.67 ± 23.6
Random Tracks	0.5	892.62 ± 41.48

Table 1: Average scores over 100 consecutive games on random racetracks. Test tracks were generated from random seeds 1 through 100.

Figure 1 compares the performance of the models trained on a single racetrack. Even though both models were trained on the same racetrack, the model with dropout performed better than the model without dropout, especially on curves that the agent did not see examples of during training. This indicates that using dropout is an effective regularizer and improves the generalizability of these models.

4 Conclusion

We demonstrated a deep RL solution to CarRacing-v0 where the agent was trained on just three fixed racetracks. Even with about the environment, the agent is capable of making informed decisions. By applying dropout we improved the algorithm's ability to generalize to situations it has not observed during training. These results suggest that widespread regularization techniques such as dropout can be applied successfully to improve robustness of deep RL algorithms. This is especially fruitful in applications particularly prone to overfitting, such as environments where the agent has access to only a subset of the state space during training.

References

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

[2] M. A. Farhan Khan and Oguz H. Elibol. Car racing using reinforcement learning. 2016.

[3] Oleg Klimov. CarRacing-v0. `https://gym.openai.com/envs/CarRacing-v0/`. Accessed: 2018-09-01.

[4] 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.

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

5 Appendix

5.1 Network architecture

See the table below for the exact architecture used in our models.

Layer	Topology	Activation
1	Convolutional, 32 8x8 kernels with stride 4	ReLU
2	Convolutional, 64 4x4 kernels with stride 2	ReLU
3	Convolutional, 64 3x3 kernels with stride 1	ReLU
4	Dense, 512 neurons	ReLU
output	Dense, 5 neurons	Linear

5.2 Acetion space

The action space of OpenAI's CarRacing environment is the set $[-1, 1] \times [0, 1]^2 \subset \mathbb{R}^3$. We discretize this space into five possible actions. See the table below for more detail.

Value	Interpretation
$[-1, 0, 0]$	Steer left
$[1, 0, 0]$	Steer right
$[0, 1, 0]$	Accelerate
$[0, 0, 0.8]$	Decelerate
$[0, 0, 0]$	Do nothing

5.3 Training parameters

See the table below for the training parameters that were used during training for our models.

Parameter	Value
Exploration rate	Decrease from 1 to 0.1 linearly over the first 250000 frames, 0.1 thereafter
Optimizer	Default Tensorflow AdamOptimizer
Learning rate	0.00025
Discount value	0.99
Batch size	32
Replay capacity	100000