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

Patrik Gerber*
University of Oxford

Jiajing Guan*
George Mason University

Elvis Nunez*
Brown University

Kaman Phamdo*
University of Maryland

Tonmoy Monsoor
University of California, Los Angeles

Nicholas Malaya†
AMD Research

Abstract

Recent successes in Reinforcement Learning (RL) models have occurred when they are trained for narrow, well-defined tasks. These models perform well in their defined task, but changes in the environment often cause disproportionate reductions in performance. This paper provides an example which suggests that regularization methods improve the generalization capacity of deep RL algorithms in a case where the agent is unable to observe the task features fully. This is explored in OpenAI's CarRacing-v0 environment where the agent observes only a small subset of the state space during training. This preliminary result indicates that more complex models solved imprecisely through regularization may have higher generalization than simple models solved exactly.

1 Introduction

Deep reinforcement learning methods have proven successful in solving well-defined computer games [8, 10, 15, 1]. 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 [5, 13]. This indicates the models are not generalizing well, have overfit to prior experiences, and are not likely to transfer to other tasks effectively [3].

In this paper, a deep RL algorithm that uses dropout [11] successfully solves the game of CarRacing-v0 [7]. The model is trained on a limited state space made up of 3 tracks used as a form of curriculum learning [2]. This result shows that regularization methods such as dropout can mitigate the overfit usually exhibited by deep RL. This task is particularly challenging because the track is randomly generated for each game. A recent solution to this environment using a generative networks, but previous attempts using deep RL methods have not been successful [4, 6]. It is believed this solution is the first to solve the challenge using deep RL.

In this work, the DDQN algorithm [9, 14] was used. Following the original DQN algorithm [9], 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. This is compared with a modified version, where a dropout layer with a dropout probability of 50% is added to the second convolutional layer to regularize the model. The models were trained with varying observability of the environment. To vary the observability of the algorithm, experiments were performed on three different subsets of the state space: a single fixed racetrack, three fixed racetracks, and randomly generated racetracks. The models were then tested on 100 random tracks.

*Denotes equal contribution to this work.
†nicholas.malaya@gmail.com

Preprint. Work in progress.

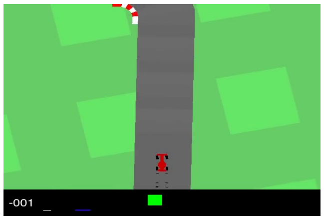

Figure 1: CarRacing-v0 environment. The red car's score increases as it traverses the track (grey cells), and decreases when leaving the track (green cells).

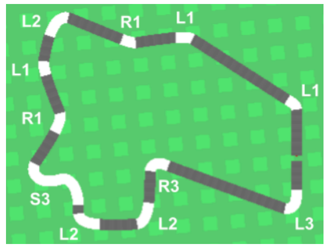

Figure 2: An example of a full track, with each corner labelled by the corresponding curve type.

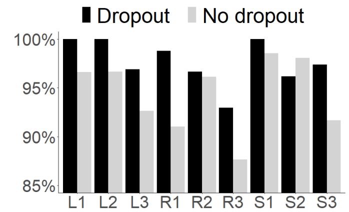

Figure 3: Percentage of tiles visited on different curves. The grey bars are the general DDQN network, and the black bars are the network with the inclusion of a dropout layer.

2 Performance Analysis

As shown in Tables 1 and 2, the average scores over 100 random games improved in all three environments when dropout was applied to the Q-network. A curve classification algorithm, demonstrated in Figure 2, was developed to investigate the generalization capacity of these models. Each curve in the racetrack is characterized 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, the percentage of tiles visited is used as a metric of the car's performance.

Figure 3 shows the percentage of tiles visited for each curve type over 100 tracks. In this comparison, both models were trained using a single racetrack, and both used the same racetrack as input data. Dropout serves to improve the car's performance through each curve, with particularly large improvements in L3, R3, and S3, the steepest curves. These steep curves are the least common curves encountered in the training data. It is possible that the unregularized models are not accurately adapting to these curves, and are overfitting to the more common (and shallower) curves. This can be catastrophic, as a sharp turn requires less velocity or the car will leave the track.

Environment	Average Score
1 Fixed Track	849.99 ± 78.72
3 Fixed Tracks	853.88 ± 127.71
Random Tracks	854.83 ± 107.14

Table 1: Performance without dropout

Environment	Average Score
1 Fixed Track	894.38 ± 24.5
3 Fixed Tracks	$\mathbf{906.67 \pm 23.6}$
Random Tracks	892.62 ± 41.48

Table 2: Performance with dropout. The game winning result is shown in bold.

In addition to a higher average score, the standard deviation of scores substantially decreased in the cases using dropout. Observations indicate that this reduction in variation was because of fewer catastrophic crashes of the car in the models regularized with dropout.

3 Conclusions and Future Work

A deep RL solution was shown to solve CarRacing-v0 where the agent was trained on just three fixed racetracks. Regularization improved the algorithm's ability to generalize to situations not observed during training. These results suggest that common regularization techniques such as dropout can be applied successfully to improve robustness and generalization capacity 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. Higher capacity networks may change the balance between optimal parallelization of RL tasks between CPUs and GPUs [12]. These results are preliminary and are only shown for the CarRacing-v0 environment. Future work will focus on extending these methods to additional environments.

References

[1] Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent complexity via multi-agent competition. *arXiv preprint arXiv:1710.03748*, 2017.

[2] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

[3] Jilles Steeve Dibangoye and Olivier Buffet. *Learning to Act in Decentralized Partially Observable MDPs*. PhD thesis, INRIA Grenoble-Rhone-Alpes-CHROMA Team; INRIA Nancy, équipe LARSEN, 2018.

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

[5] Ken Kansky, Tom Silver, David A Mély, Mohamed Eldawy, Miguel Lázaro-Gredilla, Xinghua Lou, Nimrod Dorfman, Szymon Sidor, Scott Phoenix, and Dileep George. Schema networks: Zero-shot transfer with a generative causal model of intuitive physics. *arXiv preprint arXiv:1706.04317*, 2017.

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

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

[8] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[9] 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.

[10] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[11] 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.

[12] Adam Stooke and Pieter Abbeel. Accelerated methods for deep reinforcement learning. *arXiv preprint arXiv:1803.02811*, 2018.

[13] Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685, 2009.

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

[15] Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, et al. Relational deep reinforcement learning. *arXiv preprint arXiv:1806.01830*, 2018.

4 Appendices

4.1 Car-Racing-v0

OpenAI Gym's CarRacing-v0 reinforcement learning testbed [7] 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 track. 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. The environment is considered solved when the agent achieves an average score of 900 or above over 100 consecutive games.

4.2 Implementation

The DDQN algorithm [14] was used. 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 [9], 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. The dropout probability was calibrated to 50% for maximum performance. The model was 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 best model was attained using three tracks as a form of curriculum learning. These three tracks are shown below.

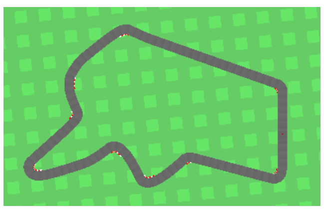

Figure 4: The first training track.

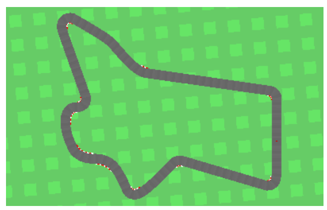

Figure 5: The second training track.

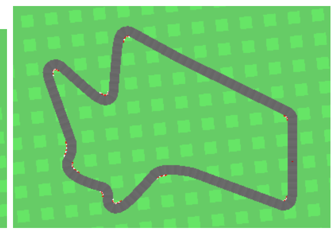

Figure 6: The third training track.

The code, implemented algorithm, and further model information can be found at `https://github.com/AMD-RIPS/RL-2018`. A video of the fully trained car running a track can be found at `https://drive.google.com/file/d/1DQU4yCsq6nbVJB6WKoXlED9YFGDselIu/view`.

4.3 Network architecture

See the table below for the exact architecture used in the 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

4.4 Action space

The action space of OpenAI's CarRacing environment is the set $[-1, 1] \times [0, 1]^2 \subset \mathbb{R}^3$. The space was discretized 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

4.5 Training parameters

The table below details the training parameters that were used during the training of the car-racing 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