Reproducibility of Gradient and Non-Gradient Based Reinforcement Learning Algorithms

Kahlan Gibson*
Electrical and Computer Engineering
University of British Columbia
kahlangibson@ece.ubc.ca

Mohamed Matar
Electrical and Computer Engineering
University of British Columbia
momran@ece.ubc.ca

Deval Shah
Electrical and Computer Engineering
University of British Columbia
devalshah@ece.ubc.ca

Abstract

This report investigates the reproducibility and results in the ICLR submission "Deep Neuroevloution: genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning" [3]. In the submitted work, a non-gradient based genetic algorithm is used to train a deep neural network for the purpose of complex deep reinforcement learning tasks. The authors state that their population-based algorithm surpasses conventional learning algorithms such deep Q-learning (DQN) and policy gradient (A3C) in terms of training time, since it reduces training time on average by 4x compared to A3C and 8x compared to DQN. Also, the authors claim that the proposed algorithm outperforms these methods in certain games. We find that while the run-times for selected experiments are representative of the published work, we were not able to reproduce the scores achieved on the Atari 2600 data set.

1 Introduction

Reinforcement learning (RL) algorithms are used for solving complex tasks such as control of autonomous vehicles [6], the game Go [13], and autonomous anesthesia machines [9]. Although the achievement of RL is promising, evaluation of published RL algorithms can be difficult to reproduce due algorithm sensitivity to hyper-parameters and insufficient information for reproducibility published in academic work [5]. Furthermore, no ideal metric exists to fairly evaluate all RL algorithms. In supervised learning, the standard practice is to split the available data set into training, validation and test sets. This practice has resulted in the rise of machine learning challenges such as ImageNet [11] and PhysioNet [1] which standardize algorithm evaluation using test data. Standardizing evaluation of RL algorithms is not as obvious, as RL isn't a supervised learning algorithm in the traditional sense. In an attempt to standardize algorithm evaluation, the Atari 2600 games in the Arcade Learning Environment [4] provide a consistent metric ("accuracy score") to compare algorithms directly. This framework, as well as training wall-clock time, and number of frames (amount of experience data), are common metrics of comparisons for algorithms in literature [8][10][12][7][3].

Despite the ubiquity of these metrics, previous work has called for more rigorous evaluation of proposed RL algorithms by reproduction of evaluation [5], prevention of result overfitting [16], and through Reproducibility Challenges[2] over concerns with RL algorithms overfitting to specific RL

*All authors contributed equally to this work

[2]ICLR Reproducibility Challenge https://reproducibility-challenge.github.io/iclr_2019/

tasks. Furthermore, wall-clock time comparisons vary widely with implementation, optimization, and targeted hardware.

In the submitted paper, Anonymous [3] applies a non-gradient-based genetic algorithm (GA) to deep RL problems to investigate the performance of this types of algorithm on deep neural networks (DNNs). The paper contribution is improved performance over "traditional" RL algorithms by utilizing novelty search to overcome sparse reward functions. Furthermore, Deep GA is more data and computationally efficient than gradient-based methods. The authors trained Atari in $\sim$4 hours on a single "modern" workstation and in $\sim$1 hour distributed across 720 cores.

The authors state that their results (1) show that genetic algorithms are suitable for large-scale DNN training, (2) suggest that gradient-based methods such as evolution strategies (ES), deep Q-learning (DQN), and policy gradient (A3C) are not an ideal solution for hard problems such as RL tasks, and (3) show improved performance through neuroevolution strategies.

In this report, we attempt to evaluate the reproducibility of the submitted paper "Deep Neuroevolution" [3], and provide a rigorous critique of the author's methodology. We perform our reproduction and evaluation through two contributions:

1. Run-time and performance evaluation by locally executing the author's genetic algorithm (GA) implementation and the baseline algorithms, available online [2]
2. Analysis and critique of locally collected results compared to results available in the paper, including reporting of implementation detail or rationale that wasn't included in the submitted paper.

2 Related Work

Reinforcement learning involves a machine actor which "learns" a behaviour policy by measuring a returned reward signal from its environment [15]. Since the agent learns through experience, both explorational and exploitative actions must be taken in order to exercise the state-action space sufficiently to develop an ideal state-action policy.

Historically, reinforcement learning algorithms such as Q-learning have been implemented using state-action space tables [15]. However, full exploration of these tables requires high levels of environment quantization to reduce input dimensionality and required training time. To avoid quantization, Q-function approximators such as neural networks are used to determine the next action for a given state. However, it has been observed that online training of reinforcement learning agents using these methods can be unstable due to temporal variance in agent experiences during training [7][14].

To overcome this instability, a method called experience replay recalls agent experiences by random sampling or batch training. However, this method requires higher memory and computational intensity. In 2015, Mnih et al. presented a method of RL agent training of deep Q-networks (DQN) [8]. In this work, a single DNN architecture consisting of 3 convolutional layers and 2 fully connected layers was trained on 50M frames of Atari 2600 games. The proposed DQN algorithm requires 12-14 days of training on GPUs to obtain scores comparable to professional human performance [10][8]. The DQN algorithm was also adapted to a massively parallel method called Gorila DQN which outperforms and runs approximately 2-10x faster than the GPU DQN algorithm, with higher performance for longer training time [10].

In 2016, Minh et al. proposed a parallelized, asynchronous gradient descent training method called asynchronous advantage actor-critic (A3C) [7] that achieved greater performance in less training time on a single 16-core CPU than methods using experience replay on specialized GPU hardware. By dispatching multiple learning agents asynchronously, this algorithm forgoes the necessity for experience replay by learning from different, parallel agent experiences [7].

Salimans et al. applied evolution strategies (ES) to reinforcement learning in 2017 [12]. This algorithm approximates gradient calculation using a method similar to finite difference approximation. Since ES forgoes the necessity for performing back-propagation or calculating a value function by calculating a score function gradient estimator using stochastic approximation, this algorithm reduces the amount of computation required by a factor of 3x. Training the ES algorithm on 720 distributed CPU cores on 1 billion (1B) game frames for an hour achieves comparable results to the published results for 1 day of A3C training on 320M frames.

Finally, the work under consideration performs DNN weight evolution by genetic algorithm (GA) [3]. The proposed genetic algorithm is a simple implementation that iteratively explores the state-space by proposing a population P of N individuals. Then, each individual is evaluated through testing to determine a *fitness* score, and the top T individuals are selected and mutated through injection of Gaussian noise to create a new population of N individuals. In order to ensure that agents both explore and exploit the training environment, the individual fitness score contains rewards for both novel actions and performance. In the work, the authors state that a 4 GPU/48 CPU workstation training a DNN using GA can achieve competitive performance in $\sim$4 hours, compared to 7-10 days for DQN and 4 days for A3C. In a highly distributed 720 CPU cores implementation, both GA and ES can train Atari in $\sim$1 hour. The authors claim that a benefit of the GA algorithm is its ability to be run on GPUs, making it possible to execute on a single workstation. Additionally, the authors observed that the GA performance did not plateau during training, and show performance after additional training (on 6B vs. 1B frames) is conducted [3].

All of the proposed attempt seek to implement stable, high-performance DNN training algorithms for reinforcement learning applications. Older work, such as DQN, Gorila, and A3C, are gradient-based methods that achieve at or above human level performance, but require days of training time. ES achieves significant wall-time improvements, but is still a gradient-based method by performing a gradient approximation calculation. The paper under consideration achieves competitive results among the collection of previous algorithms in 1-4x the time required to train ES, depending on the available workstation. The authors of Deep Neuroevolution also indicate algorithmic advantages of GA over ES: genetic algorithms don't require generational weight updates via averaging or additional forward passes for batch virtualization [3].

3 Methodology

In order to evaluate the reproducibility of the results submitted by Anonymous [3], we have chosen to perform a local evaluation of the author's code implementation available on GitHub [2]. This method has the advantage of evaluating the exact algorithm that the authors are reporting results on in their paper, and is therefore advantageous for successful reproducibility. In the interest of efficient testing, we have chosen to only attempt to reproduce the results achieved by the GA, ES, and random search (RS) algorithms, as these were most comparable in performance and run time. The RS algorithm is a naive learning agent learning algorithm that performs random actions without choosing exploitative moves during training. In the paper, the GA algorithm is shown to always outperform RS, while DQN, A3C, and ES only outperform RS in 50-75% of evaluated games [3].

Table 1 shows the summarized results presented in the Deep Neuroevloution paper.

Table 1: Q-learning (DQN), evolution strategies (ES), policy gradient (A3C), random search (RS), and genetic algorithm Atari game scores obtained in the reproduced work [3] for select games

	DQN	ES	A3C	RS	GA	GA
Frames	200M	1B	1B	1B	1B	6B
Time	$\sim$7-10d	$\sim$1h	$\sim$4d	$\sim$1h or 4h	$\sim$1h or 4h	$\sim$6h or 24h
Frostbite	797	370	191	1,164	4,536	6,220
Atlantis	279,987	1,267,410	911,091	26,371	76,273	129,167
Gravitar	473	805	304	431	476	764
Kangaroo	7.259	11,200	94	1,099	3790	11,254
Skiing	-13,062	-15,443	-10,911	-7,679	-6,502	-5,541

4 Experimental Results

By executing the available code for ES, RS, and GA on our hardware, we obtain the results presenting in Table 2.

Table 2: evolution strategies (ES), random search (RS), and genetic algorithm (GA) Atari game scores reproduced for selected games. * denotes large experiments we didn't conduct

Game	ES		GA 1B		RS		GA 6B	
	T (hr)	Score	T (hr)	Score	T (hr)	Score	T (hr)	Score
Frostbite	4.88	304	14.68	**4,220**	5.73	1,450	86.58	**8,670**
Atlantis	7.15	**171,000**	20.35	58,600	8.14	32,200	*	*
Gravitar	6.52	622	15.26	**756**	6.33	344	*	*
Kangaroo	6.81	**7,090**	16.12	6,800	7.56	1,210	*	*
Skiing	6.71	-8,990	12.25	**-6,010**	4.4	-7,210	*	*

We chose to reproduce results for a subset of the Atari 2600 benchmarks, including Frostbite, Atlantis, Gravitar, Kangaroo and Skiing. These games were chosen based on the magnitude of their scores, picking a variety of values where GA performs either very well, poorly, or somewhere in between.

Although the results presented in the Deep Neuroevloution paper were produced on specific workstation configurations (namely, a 720-core distributed CPU system and a "modern" single workstation with 4 GPUs and 48 CPUs), we are limited to reproducing these results on our available hardware. Our results were obtained by executing the ES, RS, and GA algorithms on either single-GPU or 4-GPU hardware with shared 44 CPU cores. **However, the second configuration was used for one experiment to confirm that one GPU roughly takes 4x for run-time compared to using four GPUs**.

The hyperparameters used for GA, ES and RS are kept same as given in the paper:

1. Population Size = 1000
2. Mutation Power = 0.002
3. Truncation Size = 20

In order to evaluate the genetic algorithm on extended training sets, the reproduced paper evaluated the algorithm on both 1B frames and 6B frames of training data. Even though one billion frames is chosen to be a fixed metric in all algorithms, the authors argue that it is valid to train using more data at the cost of additional training time since it is still faster than state-of-the-art techniques while having the ability to keep learning and increase the score.

We chose to reproduce the result after training for six billion frames for the **Frostbite** game only, due to constrained time and resources. The remainder of the experiments were trained on one billion frames.

For computation of the score, the highest scoring elite is selected across the generation and the elite is evaluated on 200 independent test episodes. The mean of these 200 scores is reported as *TestRewMean*. In the paper, the same experiment is repeated 5 times and median of *TestRewMeans* from all experiments is reported, but due to resource constraints, we have reported score of one experiment.

In addition, we report the training run-time for each game in each training strategy. Unlike the paper results, we found differences in run-time consumed by each game for the same algorithm which is treated as roughly the same in their results.

4.1 Performance across generations

In the paper, Number of generations required for 6B are reported. However, due to workstation limitation, we have reproduced the result with 6B frames for Frostbite only. Number of generations required for 6B frames in Frostbite is 960 which is in the range given in the paper (889-1154)

Table 3: Number of generations required for genetic algorithm training for 1B frames across different Atari benchmarks

Game	No. of Generations
Frostbite	171
Atlantis	135
Gravitar	292
Kangaroo	132
Skiing	114

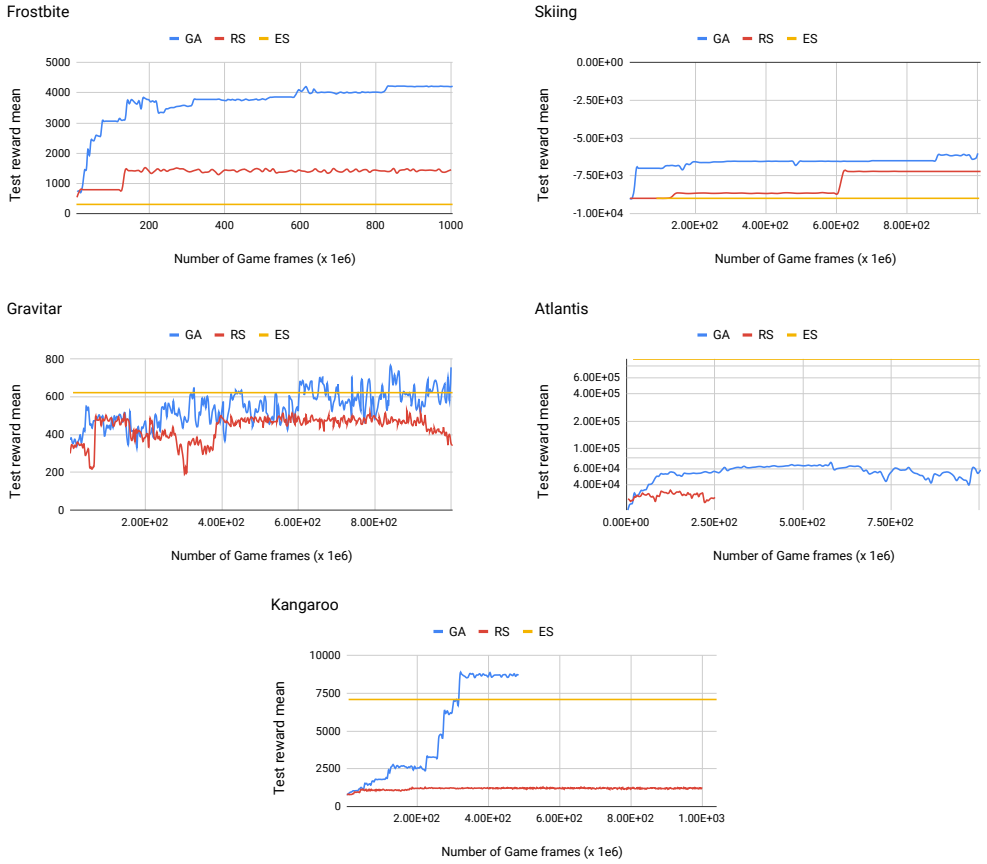

Figure 1: GA and random search performance across generations on Atari 2600 games. Here the score is the mean score of the elite over 200 test episodes. For ES, final score is plotted.

5 Discussion

In this section, we discuss our observations and analysis of the presented contribution in terms of training run-time and learning score. For our first observation we see differences in the scores we generated locally when compared to what is presented in the paper. Figure 3 shows the nominal error which varies for each game, however, certain scores vary a lot such as Kangaroo RS score. We can't justify the big differences in scores since the authors work doesn't clearly state their metric for computing the score, which we present as a reproducibility limitation for the paper.

Conversely, the training run-time we observed locally was observed to be approximately the same after taking differences in workstation hardware into consideration (one GPU vs. four GPU workstation). In order to compute the experiment run-time we would expect on hardware similar to the author's, we

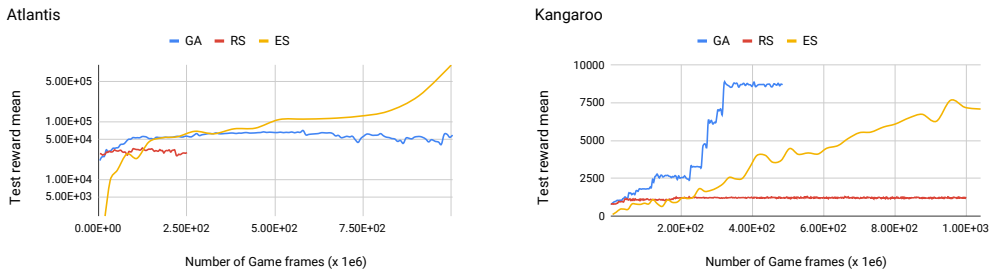

Figure 2: Score across generations with ES performance trace. ES score has not plateaued after 1B frames which suggest that ES might outperform GA for higher number of frames.

performed an experiment measuring run-time on 4 GPUs and another experiment measuring run-time on one GPU. The run-time on four GPUs was one-fourth the run-time for one GPU. Hence, we can assume linearity in training run-time vs. number of GPUs.

Finally, there are some opportunities for improvement for the authors of the submitted paper in order to make the results more easily understood and reproduced:

- The score results presented by the paper do not clearly indicate whether the scores are below, meet, or exceed the expectation based on opportunities and limitations of non-gradient-based training algorithms.
- The validity of comparing training scores using 6B frames against 1B frames should be more thoroughly explored to indicate whether score improvements are proportional to the additional run-time overhead.
- The authors claim that training score continues improving with additional training frames, while these scores were saturated for some games after training using 1B frames.
- The median of scores over 5 experiments is compared with other methods. However, variance of the scores is not reported or discussed in the results section.

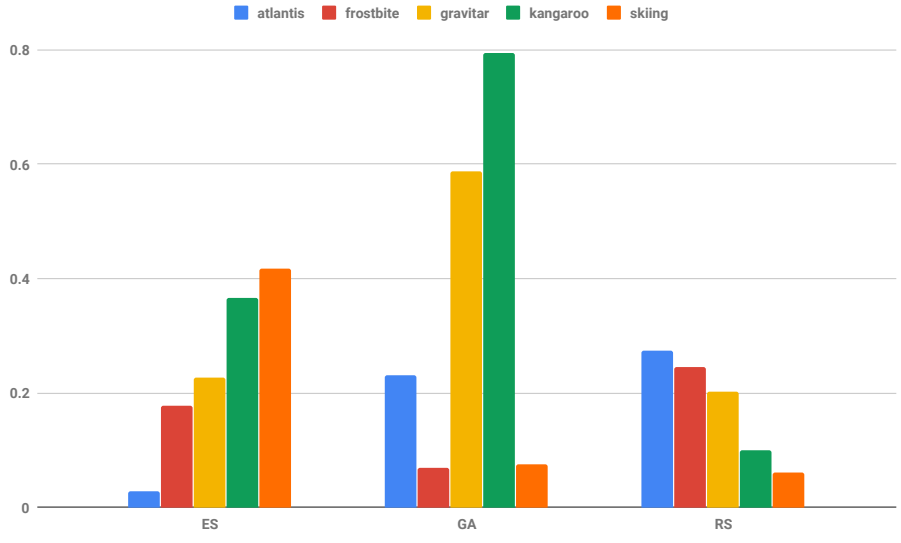

Figure 3: Nominal error between measured score and proposed score

There are also some opportunities for optimization left for future work. For example, (1) Our experiments had low GPU utilization due to inefficient parallelization of the genetic algorithm implementation. Improving parallelization for GPUs could significantly improve training run-time.

Also, (2) using more advanced GA algorithm features for learning could result in better exploration of the state-action space. The genetic algorithm used in the paper is quite simple. For example, this method evolves generations by injecting Gaussian noise into the weights of the individuals with the greatest fitness scores. A more complex implementation of genetic algorithms involves crossover of features between individuals in addition to random mutations.

6 Conclusion

In this report, we attempted to reproduce and critique the limitations of the submitted paper "Deep Neuroevolution". In the submitted paper, the authors proposed Deep GA, a method for training deep neural networks for complex reinforcement learning tasks. Using the same network topology as previous state-of-the-art gradient-based RL training algorithms, the authors claim that their algorithm achieves competitive scores on the Atari 2600 benchmarks in only $\sim$1-4 hours of training, with opportunity to train on additional data to achieve higher scores.

Our experiments attempted to reproduce these results locally on a subset of the Atari 2600 training set using available hardware resources. From our experiments, we observe significant variance between our scores and the scores reported in the paper. However, we observe that when accounting for differences in workstation hardware, our experiments required the same amount of run-time as the times reported in the paper.

As a final remark, we observe that there is still opportunity to improve the proposed genetic algorithm. For example, improving GPU utilization and exploring more complex GA algorithms may improve run-time and efficacy of state-space exploration. In order to ensure verifiable reinforcement learning results are published, a critical analysis of the reproducibility of works and their limitations should be conducted. We believe that we have achieved this goal in our project.

References

[1] Physionet/computing in cardiology challenges. `https://www.physionet.org/challenge/`, 2000-2018.

[2] Anonymous. Deep neuroevolution. `https://github.com/uber-research/deep-neuroevolution`, 2018.

[3] Anonymous. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. In *Submitted to International Conference on Learning Representations*, 2019. Under review.

[4] Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *J. Artif. Int. Res.*, 47(1):253–279, May 2013.

[5] Khimya Khetarpal, Zafarali Ahmed, Andre Cianflone adn Riashat Islam, and Joelle Pineau. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. In *International Conference on Learning Representations Reproducibility in Machine Learning Workshop*, 2018.

[6] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015.

[7] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1928–1937, New York, New York, USA, 20–22 Jun 2016. PMLR.

[8] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[9] Brett L. Moore, Larry D. Pyeatt, Vivekanand Kulkarni, Periklis Panousis, Kevin Padrez, and Anthony G. Doufas. Reinforcement learning for closed-loop propofol anesthesia: A study in human volunteers. *J. Mach. Learn. Res.*, 15(1):655–696, January 2014.

[10] Arun Nair, Praveen Srinivasan, Sam Blackwell, Cagdas Alcicek, Rory Fearon, Alessandro De Maria, Vedavyas Panneershelvam, Mustafa Suleyman, Charles Beattie, Stig Petersen, Shane Legg, Volodymyr Mnih, Koray Kavukcuoglu, and David Silver. Massively parallel methods for deep reinforcement learning. *CoRR*, abs/1507.04296, 2015.

[11] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[12] Tim Salimans, Jonathan Ho, Xi Chen, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. 2017.

[13] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, January 2016.

[14] Richard S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems 8*, pages 1038–1044. MIT Press, 1996.

[15] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[16] Chiyuan Zhang, Oriol Vinyals, Rémi Munos, and Samy Bengio. A study on overfitting in deep reinforcement learning. *CoRR*, abs/1804.06893, 2018.