Deep Reinforcement Learning in a 3-D Blockworld Environment

Trevor Barron **Matthew Whitehead** **Alan Yeung**
Colorado College
Department of Mathematics and Computer Science
{trevor.barron, matthew.whitehead, alan.yeung}@coloradocollege.edu

Abstract

Deep reinforcement learning is an effective method for training autonomous agents to a high level of performance on visual tasks. This work explores how reinforcement learning agents using deep Q-networks perform when visually processing 3-D virtual environments and how deeper network architectures can improve performance given the added difficulties of more complex environments. We provide results for tests on a variety of tasks in a virtual 3-D world and show that deeper convolutional neural networks lead to increased performance.

1 Introduction

Reinforcement learning has a long history of use in game playing. In the 1950's, Samuel, in one of the first successes of artificial intelligence research, created a program that learned to play checkers competitively [Samuel, 1959]. In the 1990's, Tesauro introduced a system that learned to play backgammon at a very high level [Tesauro, 1995]. More recently, Mnih et al. developed an algorithm called deep Q-learning that allowed agents to use raw pixel data as input and perform at a high level on a variety of Atari 2600 games available in the Arcade Learning Environment [Mnih *et al.*, 2013], [Mnih *et al.*, 2015], [Bellemare *et al.*, 2013].

Deep Q-learning [Mnih *et al.*, 2015] is a variation of Q-learning that uses deep convolutional neural networks to estimate an agent's value function, we call such a network a deep Q-learning network (DQN). Convolutional neural networks (CNNs) were first proposed decades ago [Fukushima, 1979], [LeCun *et al.*, 1989] and since have been shown to have strong performance on many computer vision tasks [Krizhevsky *et al.*, 2012], [Szegedy *et al.*, 2014]. One major benefit of this approach is that agents trained with deep Q-learning perform well without requiring any human handcrafted features to be successful. Deep convolutional networks allow the agents to learn visual features directly from the environment, resulting in increase representational capability and improved decision making.

In this work, we are interested in the performance of agents that use deep Q-learning to learn behaviors of various complexities in three dimensional environments. We are also interested in testing much deeper network architectures to see if they have a substantial effect on performance. For our experiments, we use a virtual block world inspired by the popular video game, Minecraft [Microsoft, 2016]. The use of this kind of environment for AI tasks was also proposed recently by [Aluru *et al.*, 2016]. [Tessler *et al.*, 2016] also use a similar environment for testing agents with Deep Skill Networks on lifelong learning tasks.

Our work has two main contributions. First, we present a 3-D framework that is a modification of an existing open source project [Fogleman, 2016] that easily allows for experimentation with learning behaviors of various complexities.

The framework defines a `Task` interface that makes it simple to implement and test new agents with different environments, action sets, and reward functions. Second, we show test results for DQN agents on a series of 3-D tasks of varying complexities. The agents were able to perform well on a variety of tasks including visually targeting certain blocks, navigating along raised walkways of various widths, and collecting blocks in a large, complex world. We show that the relatively shallow architecture from [Mnih *et al.*, 2013] provides good performance for simpler 3-D tasks, but deeper architectures yield a substantial performance gain for more complex tasks.

2 Environment Description

The environment we use to perform our experiments is a 3-D virtual world comprised entirely of blocks of various types, much like the popular game, Minecraft [Microsoft, 2016]. This environment is similar to Microsoft's Project AIX framework mentioned in [Abel *et al.*, 2016]. The agent operating in the virtual world has a first-person perspective when receiving visual environmental input. Each tick of the game, the agent receives an 84x84 screenshot of its view of the world. This is the only input the agent receives: it has no other built-in model of the layout of the world. The agent must then choose an action from the set of possibilities: *rotate view up, rotate view down, rotate view left, rotate view right, step forward, step backward, step left, step right, jump forward, and try to break a block*. Given the fine-grained movements and the large, complex virtual environments, the number of possible states of the game is very large.

When the agent chooses an action to look in a certain direction, the agent's view rotates by a small, fixed number of

degrees in that direction (about 2 degrees). When the agent takes a step in one of the four possible directions, the step size is also a small, fixed amount (about 1/4 of a block unit). If the agent chooses the action to break a block, then the block in the center of the field of view, if it is within eight blocks, is removed from the environment. There are three types of blocks in our experiments: grass (brown on the bottom and green on top), bricks (reddish colored), and stone (grey colored). The grass and brick blocks are breakable, but the stone blocks are not. The agent is able to jump slightly higher than a single block, so it is possible for it to jump on top of a block directly in front of its view. Figure 1 shows an example screenshot from an agent's view during one of our experiments.

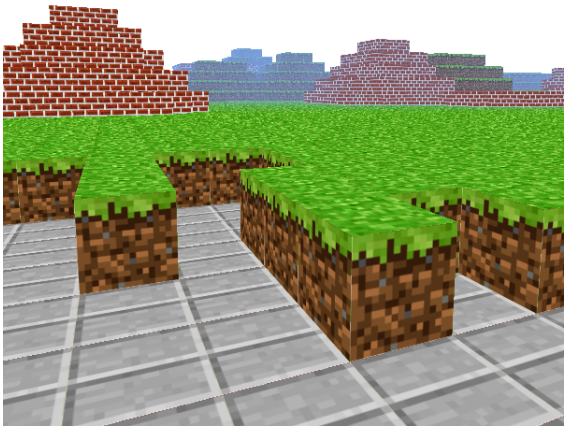

Figure 1: Example 3-D blockworld environment. The bottom layer consists of stone blocks, the layer above the stone is grass, and there are hills of brick and grass in the distance.

3 Description of Models

We tested two network architectures with different numbers of convolutional layers in the environment described above. The first architecture follows [Mnih *et al.*, 2013] with two convolutional layers (see Figure 2).

The second network follows the VGG network of [Simonyan and Zisserman, 2014] with a much deeper architecture consisting of 16 convolutional layers (see Figure 3). The convolutional layers all have a similar parameter structure with a set kernel size of three and stride of one. We chose the second model architecture in order to see if a network with a deeper structure that had been shown to be more successful on standard computer vision datasets, such as ImageNet [Russakovsky *et al.*, 2015], would be helpful for DQN agents learning more complex tasks in more complicated 3-D worlds.

The network with two convolutional layers had roughly 270,000 weights while the network with 16 convolutional layers had nearly 17 million weights (nearly 63x as many

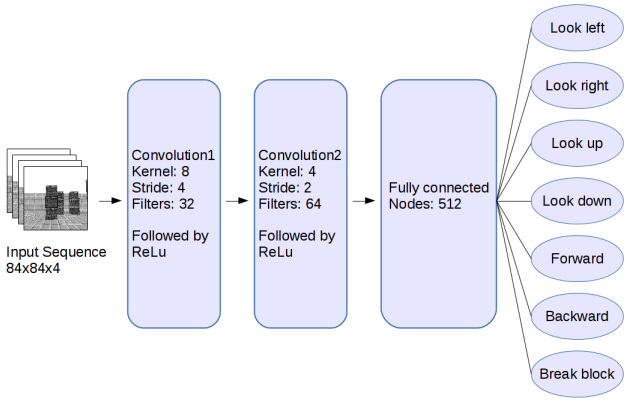

Figure 2: The first test architecture with two convolutional layers.

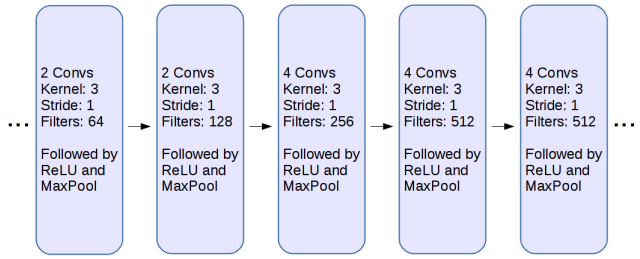

Figure 3: The convolutional section of the second test architecture with a total of 16 convolutional layers. This section is followed by two fully-connected layers with 4096 nodes each (each with 50 percent dropout on their outputs).

weights as the shallower network). We were interested to see if the larger network would be more capable of representing the larger state spaces of more complex 3-D tasks.

4 Experiments

4.1 World Depth Estimation

For the agents to be successful in the relatively complex 3-D world, they would have to have some ability to determine the distances of obstacles based on their first-person perspective views. Determining distance in our 3-D virtual environment is similar to the problem of real-world monocular depth estimation from computer vision [Saxena *et al.*, 2005], [Liu *et al.*, 2014], [Chen *et al.*, 2016]. The two questions we wished to answer were, *Can an agent with a deep neural network-based policy determine the distance of the block it is targeting to a very high degree of accuracy?* and *Can that agent also accurately determine the distance to all the visible blocks in the world?* We reasoned that if the network could accurately predict depth, then a DQN with a sufficient number of parameters could do the same. Such an agent would then be well-suited for behavioral tasks requiring spatial information. Since the DQN will need to represent more spacial information than in a 2-D task, we also examined whether network architectures with additional convolutional layers resulted in

more accurate depth estimations for both the agent's center-of-view and for all visible pixels.

First, we ran an agent that made random moves in a randomly generated blockworld environment and recorded screenshots of the agent's view, along with the distance in blocks from the agent to the object at each pixel. This distance label was determined directly from the OpenGL model of the world. Blocks further than 50 block units away from the agent were given the label of 50. The dataset labels were then normalized to the range $[0.0, 1.0]$.

We ran two tests to determine the ability of the network to learn depth information. We trained the model architectures described above to convergence in the Keras Deep Learning Library [Chollet, 2015]. We found that both networks converged well when estimating the single center pixel distance (Figure 4) and the distance to all the pixels (Figure 5). The graphs show the normalized mean squared errors for the distance predictions and Table 1 shows the final errors for both of the tested architectures when translated into block units. The deeper network outperformed the shallow network in both the experiments, but the margin was small (12% improvement and 4.5% improvement for the two tasks, respectively).

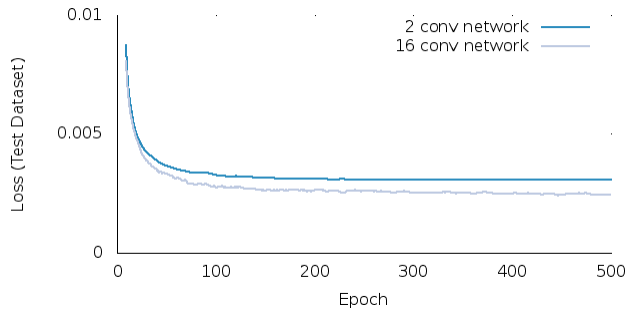

Figure 4: Error (MSE on normalized block distances) on test dataset when estimating pixel distance for the center pixel of the agent's view.

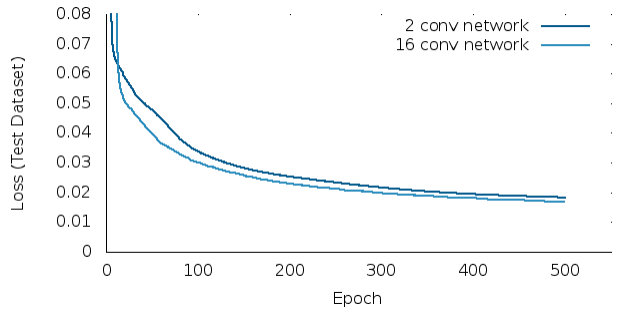

Figure 5: Error (MSE on normalized block distances) on test dataset when training the network to estimate pixel distance for all the screenshot pixels.

Figure 6 shows several example screenshots from the depth test along with ground-truth, prediction, and error map images. From the error map images we can see that the model had difficulty with pixels that lie near the boundary between block structures and the sky (since there are many light-colored pixels near these boundaries) because a one pixel change in the view would drastically change the target distance.

Architecture	Single Pixel	All Pixels
2 Conv Layers	2.77	6.78
16 Conv Layers	2.44	6.48

Table 1: Network distance estimation errors (in block units)

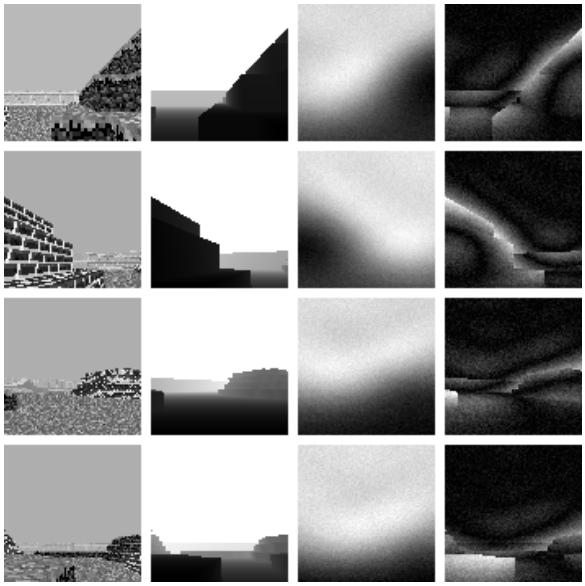

Figure 6: Four examples from the depth estimation test. The first column shows the original greyscale screenshots. The second column shows the ground-truth distance images (darker pixels are closer and lighter pixels are further away from the agent). The third column shows the model's predictions for all the pixel distances. The final column shows an error map of the absolute differences between the ground-truth and model values for each pixel (lighter values are larger errors).

The fourth row in Figure 6 also shows a screenshot for which the model failed to recognize a nearby block in the lower-left corner of the screen since its top texture was so similar to the ground further ahead. This mistake is clearly visible in the error map image in the last column (there is a bright white shape in the lower-left corner that is the top of an unrecognized block).

In conclusion, our experiments show that these networks can determine block distances to a high degree of accuracy, meaning they may be well-suited for 3-D tasks. In addition, the deeper network performed slightly better than the shallow network when estimating pixel distances indicating that it has the potential to perform better on more difficult spatial tasks described below.

4.2 Blockworld Tasks

To determine the performance of the DQN agents, we tested them on a variety of behavioral tasks in the previously-described 3-D environment. Each of these tasks is detailed below and the hyperparameters for all tasks are given in Table 2.

Our implementation uses the Caffe framework [Jia *et al.*, 2014] and the base system is a modification of an existing open-source project [watts4speed, 2016]. We added an interface for the blockworld environment to perform our experiments and our code is available at: https://github.com/tpbarron/fast-dqn-caffe/tree/minecraft

Parameter	Value
Replay memory size	500,000
Pre-learning steps	100
Reward values	{-1, 0, 1}
Minibatch size	32
Gradient clipping threshold	10
Learning rate	0.1
Future reward discount	0.95
Learning algorithm	AdaDelta

Table 2: Experiment Hyperparameters

Simple Block Targeting

Our first task involved breaking floating grass blocks randomly placed in the air around an enclosed room with dimensions 50x50 blocks. The agent had a limited number of actions to break as many grass blocks as possible. Each broken grass block yielded a positive reward, broken brick blocks yielded a negative reward, and all other actions resulted in no reward. Each test room also included floating brick blocks that were intended to make the task slightly harder since they were in similar positions to the floating grass blocks. To achieve the highest cumulative reward in the end, the agent would have to learn to successfully navigate the room and rotate its view to target the correct floating blocks. Figure 7 shows an agent's view during this task.

Floating Walkway

In the second task, the agent was required to make its way as far as possible along a floating, meandering walkway. If the agent moved off the walkway, it would then fall and the episode would end. The agent was given a reward each time it made it further down the walkway than it had ever been before in that episode. Every episode a random walkway of length 100 was generated. We tested two different versions of this task: one with walkways of width five blocks (wide walkway) and one with walkways of width one (narrow walkway). Figure 8 shows an example screenshot of an episode with the narrow walkway.

Block Collecting in a Complex World

The last test we performed involved a large, complex world with many blocks (roughly 100,000) and varying terrain, including hills comprised of different block types. For this test, the agent was rewarded when breaking grass blocks, which were abundantly available, but not rewarded when breaking brick blocks, which were also widely available. There were many blocks to break, but the agent was given a limited number of actions to perform in each episode, so it had to prioritize actions to achieve the highest ending cumulative reward even if it encountered states with large values of short-term rewards. Because of the complex terrain, it was also important for the agent to be able to navigate effectively without getting stuck. Figure 1 shows a screenshot of the world used for this task.

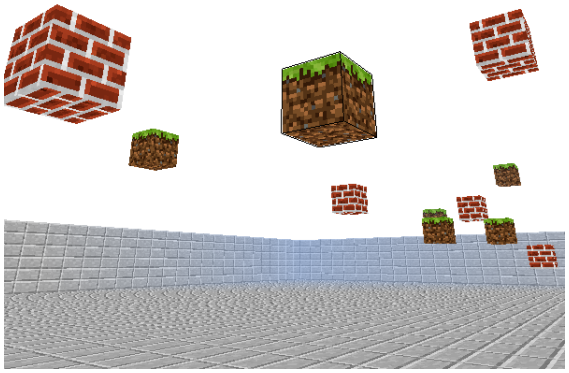

Figure 7: Screenshot from the Simple Block Targeting task. The agent is rewarded for targeting the floating grass blocks.

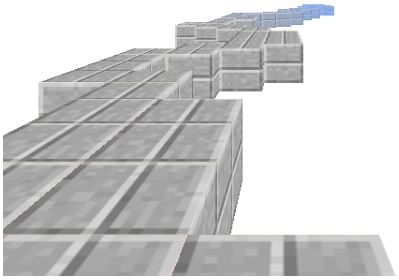

Figure 8: Screenshot from the Floating Walkway task. The agent gets rewarded for making forward progress down the walkway.

5 Results

Sample executions of each of the trained agents can be viewed in this demonstration video:

https://www.youtube.com/watch?v=6jlaBD9LCnM

All the agents were trained on a single NVIDIA Titan X GPU. The shallow network performed well on the first two tasks: Simple Block Targeting and Block Collecting in a Complex World. Figures 9 and 10 show the agents' average epoch scores during training, where each score is the average cumulative reward for an epoch.

The walkway tasks proved more difficult for the shallow network. Figures 11 and 12 show that the deeper network

outperformed the shallow network by a substantial margin. We reason that the deeper network was able to detect very small differences in the environment state from the screenshot more precisely than the shallow network. These differences were especially important in the walkway tasks because of the finality of making an incorrect decision, the agent falling, and ending the episode.

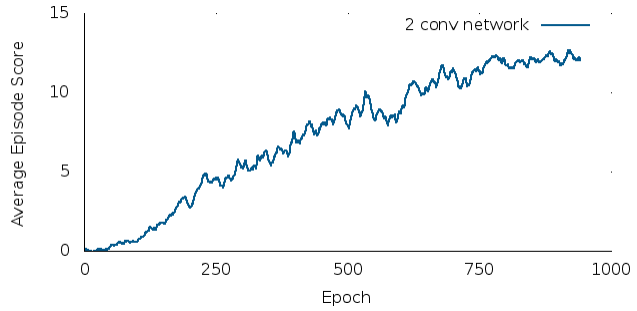

Figure 9: Average episode scores for the Simple Block Targeting task. The maximum possible score is 25 (one for each of the target grass blocks), but it was often not possible to achieve given the agent's limited number of game actions.

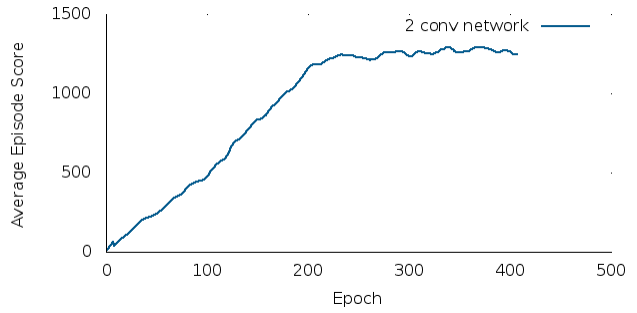

Figure 10: Average episode scores for the Block Collecting in a Complex World task. Each grass block yields 1 point and each environment has tens of thousands of grass blocks, but the agent was limited to 5000 actions per episode.

6 Conclusion

In this work, we presented a 3-D blockworld environment for testing deep reinforcement learning agents. We then showed results from experiments of pixel distance estimation and reinforcement agent performance on a variety of tasks using two different network structures. We found that when using the shallow network architecture with two convolutional layers, agent performance is good for the two easier tasks. The more difficult walkway experiments showed that there is a benefit to using a deeper network structure such as the 16-convolution VGG architecture.

In the future, we hope to test other network architectures with DQN agents in our environment. We are also interested in exploring the relationship between the number of states relevant to a task and the number of network parameters required to learn that task effectively.

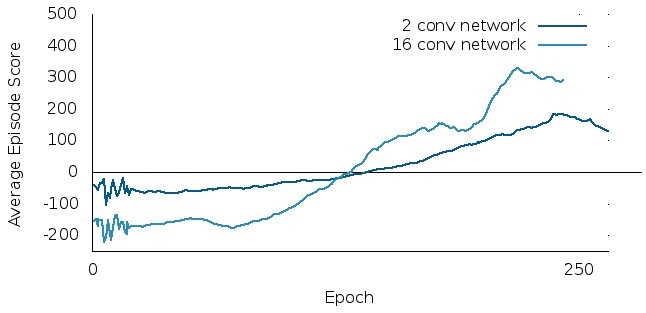

Figure 11: Average episode scores for the Floating Walkway task with walkway width=5. The maximum possible score for each episode was approximately 450 points.

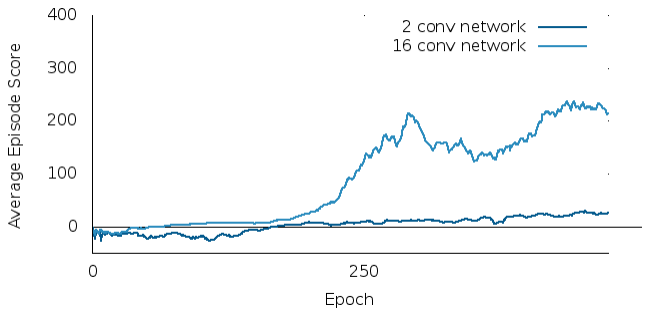

Figure 12: Average episode scores for the Floating Walkway task with walkway width=1. Similarly to the wide walkway task, the maximum possible score for each narrow walkway episode was approximately 450 points.

7 Acknowledgments

We gratefully acknowledge the NVIDIA Corporation for the donation of the Titan X GPU used to support this research.

References

[Abel *et al.*, 2016] David Abel, Alekh Agarwal, Fernando Diaz, Akshay Krishnamurthy, and Robert E. Schapire. Exploratory gradient boosting for reinforcement learning in complex domains. *CoRR*, abs/1603.04119, 2016.

[Aluru *et al.*, 2016] Krishna Aluru, Stefanie Tellex, John Oberlin, and James MacGlashan. Minecraft as an experimental world for ai in robotics. *AAAI Fall Symposium*, 2016.

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

[Chen *et al.*, 2016] W. Chen, Z. Fu, D. Yang, and J. Deng. Single-Image Depth Perception in the Wild. *CoRR*, abs/1604.03901, 2016.

[Chollet, 2015] Franois Chollet. Keras. https://github.com/fchollet/keras, 2015.

[Fogleman, 2016] Michael Fogleman. Minecraft-inspired game. https://github.com/fogleman/Minecraft, 2016. Accessed: 2016-04-07.

[Fukushima, 1979] K. Fukushima. Neural network model for a mechanism of pattern recognition unaffected by shift in position - neocognitron. *Trans. IECE*, J62-A(10):658–665, 1979.

[Jia *et al.*, 2014] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS: Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 1106–1114, 2012.

[LeCun *et al.*, 1989] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Human-level control through deep reinforcement learning. *Neural Computation*, 1(4):541–551, 1989.

[Liu *et al.*, 2014] Fayao Liu, Chunhua Shen, and Guosheng Lin. Deep convolutional neural fields for depth estimation from a single image. *CoRR*, abs/1411.6387, 2014.

[Microsoft, 2016] Microsoft. Minecraft video game. https://minecraft.net/, 2016. Accessed: 2016-01-31.

[Mnih *et al.*, 2013] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[Mnih *et al.*, 2015] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:1529, 2015.

[Russakovsky *et al.*, 2015] 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.

[Samuel, 1959] Arthur L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):211–229, 1959.

[Saxena *et al.*, 2005] Ashutosh Saxena, Sung H Chung, and Andrew Y Ng. Learning depth from single monocular images. In *Advances in Neural Information Processing Systems*, pages 1161–1168, 2005.

[Simonyan and Zisserman, 2014] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[Szegedy *et al.*, 2014] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[Tesauro, 1995] Gerald Tesauro. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3), 1995.

[Tessler *et al.*, 2016] Chen Tessler, Shahar Givony, Tom Zahavy, Daniel J. Mankowitz, and Shie Mannor. A deep hierarchical approach to lifelong learning in minecraft. *CoRR*, abs/1604.07255, 2016.

[watts4speed, 2016] watts4speed. fast-dqn-caffe. https://github.com/watts4speed/fast-dqn-caffe, 2016.