# An In-Depth Investigation Of The Effects Of Feature Reduction On The Performance Of DQN Atari Agents

Holt Spalding
Oliver Newland

11/8/18

## 1 Project Overview

In this paper, we would like to investigate the effects of image preprocessing on the overall performance of various Atari agents. When training an RL agent with a DQN, you are always faced with a tradeoff: to what extent should I compress my training images in order to save computational time without losing crucial aspects of the original image? Downsampling, frame skipping (not image preprocessing, but in the same spirit), gray scaling, and image cropping have all proven effective methods of dimensionality reduction in Atari DQN, however no one has seemed to study these methods in any significant depth. We would like to study these preprocessing techniques at their most extreme in order to answer the question...how compressed can we get? For example, while reducing image resolution to an almost unrecognizable extent may result in more timesteps to convergence for a Pacman agent, is there a chance that wall clock time to convergence would be reduced? By comparing the performance of agents across these two measures (number of time steps and real time to convergence), we would like to develop a method by which other researchers can quickly determine the optimal level of image preprocessing for their given task. Furthermore, we would like to rank these different preprocessing techniques in terms of effectiveness. For example in the context of Pacman, gray scaling probably has little negative effect on performance. Therefore, gray scaling could be ranked as an essential aspect of image preprocessing when training a DQN Atari agent. Other image preprocessing techniques however may pose more tradeoffs.

We would like to conduct this experiment with something like the Pacman environment because it is deterministic, and therefore could be learned completely blind. In this experiment we would like to compare the relative performances of a blind agent, an agent that sees the state space in full resolution, and agents with varying levels of state-space visibility in between. Performance will be measured in terms of the two measures of time convergence mentioned above, as well as susceptibility to environmental noise. Resilience to environmental noise is a crucial aspect of training an effective RL agent. Since a blind Pacman agent would learn to play the game the same way every time, it would stand no chance of surviving if a new ghost NPC were to suddenly enter the environment. By the end of our experiment, we would like to optimally reconcile the various tradeoffs posed by image preprocessing in order to find the agent that performs the best in terms of time steps to convergence, real time to convergence, and resilience to environmental noise.

To summarize, these are our primary goals:

- Aim 1: Explicitly layout the costs and benefits of different image preprocessing techniques on DQN Atari agent performance.
- Aim 2: Optimally reconcile the various tradeoffs posed by image preprocessing in order to find the Pacman agent that performs the best in terms of time steps to convergence, real time to convergence, and resilience to environmental noise.
- Aim 3: Provide a framework by which other researchers can quickly determine the optimal level of image preprocessing necessary for their experiment.

## 2 Background and Related Work

The following articles, papers, and web sources provide information used in the formation of our experiment, as well as guidance for executing the experiment properly, particularly in designing it for a DQN.

### 2.1 "DQN: Does it scales?[sic]" by Harsh Satija [1]

McGill master's student Harsh Satija provides a useful outline of employing a Deep Q-Network in reinforcement learning problems. DQN's are at the forefront of reinforcement learning as they highly powerful and more widely applicable than CNN, RNN, and LSTM methods, which are locked into certain sample spaces and are liable to diverge. DQNs avoid these pitfalls by implementing "experience replay," which purges random correlations to smooth data distribution, updating action values iteratively, and using a separate target network that is updated every so often. They are limited in that they only act upon that last four observed states, and require a great deal of computational power. It is this tradeoff in performance and resource usage we are trying to explore and optimize.

## 2.2 "Playing Atari with Deep Reinforcement Learning" by Volodymyr Mnih *et. al.* [2]

Mnih and company present their groundbreaking model for learning control policies from "high-dimensional sensory input" in a reinforcement learning setting. They use a DQN method to surpass previous RL performance in a collection of seven Atari games, and surpass human performance in three of them. This article provides a strong baseline to compare our results too, and also briefly discusses pixel downsampling, one of the feature compressions we want to explore.

## 2.3 "Dynamic Frame skip Deep Q Network" By Aravind S. Lakshminarayanan *et. al.* [3]

Lakshminarayanan and company explore the possibilities of varying the frame skip rate when learning Atari 2600 games with a DQN algorithm. In their words: "A frame skip value of $k$ allows the agent to repeat a selected action $k$ number of times." More specifically, the learning agent only receives the state every $k$ frames, and its actions are carried out for intervals of $k$ frames. For states that resemble images, such as the ones from Atari 2600 games, this helps save computational resources if done well. Lakshminarayanan *et. al.* found that values of $k$ can be varied to affect outcome, and this variation can be performed by the a learning algorithm itself. Although we will not mimic this dynamic tuning, the paper is nevertheless helpful in implementing $k$ as an experimental variable.

## 2.4 "Frame Skipping and Pre-Processing for Deep Q-Networks on Atari 2600 Games" by Daniel Seita [4]

Seita provides a detailed guide to working with the Arcade Learning Environment library [5] and the open-source DQN deep_q_rl [6] used by Mnih and company in their aforementioned article. He includes several of his pitfalls and how he overcame them, which is particularly useful for getting started.

## 2.5 deep_q_rl by Nathan Sprague [6]

An open source DQN based on Google's open-source, but harder to use and understand, DQN. It seems to provide identical functionality, with the addition of easier parameter tuning.

## 2.6 Open AIGym [7]

One possible source for emulating Atari 2600 games.

### 2.7 Arcade-Learning-Environment by Marc Bellemare [5]

Another possible source for emulating Atari 2600 games, built on top of the Stella emulator. Very similar to Open AIGym, to the point where we do not currently know which to use.

## 3 Problem Formulation and Technical Approach

As described above, (aside from the blind agent) we would like to vary our DQN agents in terms of 4 features: grayscale or not, level of downsampling, number of frames skipped between each analyzed frame, and the level of cropping. We plan to test our agents on different games provided by the Open AI Atari environment, Pacman being our main focus. We also plan to use a CUDA card that will allow us to effectively measure the real time GPU performance of our different agents. We would like to use tensorflow for this project since we are relatively familiar with it and we think it will give us effective control of the underlying hardware.

Other than variations in these 4 features, all agents will be designed based almost entirely on the agents described in the 2013 Deep Mind paper we read in class. The blind agent will be an MDP designed similarly to these other agents, however it will have a significantly reduced state space in which actions are chosen on the basis of position relative to its starting position. It will know its position relative to its starting position and the number of steps it is into the game by incremental update after each action it performs.

## 4 Evaluation and Expected Outcomes

Our experiment will measure the performance of the agents with differing "visibilities" by measuring GPU cycles and wall clock time to capture both computational resources used by the DQN and overall time to convergence. Following those results, we are interested in the agents performances in response to noise. In other words, having learned on a deterministic game (Pac-Man), what resolution is required to make the adjustments needed to win?

We expect that the less an agent can see, the less resources it will use, and that we will discover an optimal compression level. We postulate that the totally blind agent will be able to learn the game relatively quickly by simply learning and memorizing the optimal path, taking more episodes, but doing so without running a DQN iteration every action. Following the introduction of noise, we expect that optimal compression level agent will still perform well, but the more blind agents will perform significantly worse.

We extend this expectation to other games in that the more randomness involved in the games rewards and obstacles, the more visibility is necessary to train an effective and adaptable agent.

## 5 Stretch Goals

We would like to discuss with you potential ways to make this project more interesting. For example, are there ways an image could be affected that improve performance than no one has tried before? One thing we wanted to look into potentially was using partial convolutions of an image to speed up DQN.

## References

[1] Harsh Satija. Dqn: Does it scales? Unpublished paper.

[2] 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.

[3] Aravind S. Lakshminarayanan, Sahil Sharma, and Balaraman Ravindran. Dynamic frame skip deep Q network. *CoRR*, abs/1605.05365, 2016.

[4] Daniel Seita. Frame skipping and pre-processing for deep q-networks on atari 2600 games, 2016.

[5] Marc G. Bellemare. Arcade-learning-environment. https://github.com/mgbellemare/Arcade-Learning-Environment, 2018.

[6] Nathan Sprague. deep_q_rl. https://github.com/spragunr/deep_q_rl, 2016.

[7] Open aigym.