INTRODUCTION

Super Mario Bros is a 2-D platformer game released for the NES in 1985. The objective is to complete levels by traversing the terrain rightward without dying. My program is built on a NES learning environment based on Google's ALE for Atari games.

ALGORITHM

Mario uses Deep Q-learning with experience replay where we optimize the loss function with stochastic gradient descent. After performing experience replay, in which we apply Q-learning updates to a randomly selected batch of recent experiences. Afterward the agent selects and
executes an action according to an ε-greedy policy.

INPUT

The NES console outputs 256x224 pixels and 48 colors. The image is down-sampled and cropped to use 84x84 grayscale pixels in order to reduce computational complexity. Additionally, given that motion is a key component on the game, each state consists of the most recent 4 frames displayed.

NEURAL NETWORK

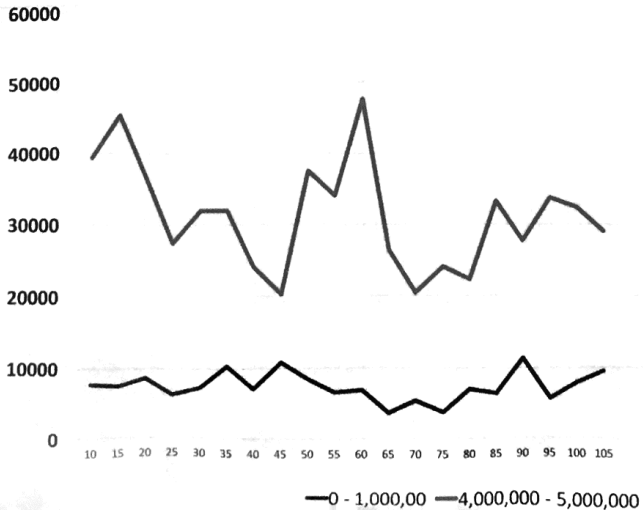
Super Mario has the potential for an extremely large state/action space. The computational impact of this was mitigated by using convolutional neural networks for function approximation

First Layer: 32 8x8 filters with 4 pixel stride followed by a rectifier non-linearity
Second Layer: 64 4x4 filters with 2 pixel stride followed by a rectifier non-linearity
Third Layer: 64 3x3 filters with 1 pixel stride
Fourth Layer: Fully Connected 512 rectifier units
Output: Predicted Q values of each of possible action

RESULTS & OBSERVATIONS

I iterated through many versions of the neural network layers, and found the structure shown to provide the best, though further tweaks could be done to continue improving the agents performance. When I began, I found that Mario's scores would increase after a relatively low number of steps, and then begin to gradually decrease with further training. Also when I began, my reward was based solely on the change in Mario's score. However, I found this led to Mario dying frequently, even after long period of training, so I added a negative reward for dying. Just recently, I was able to use my CUDA-equipped GTX 1080 to train, after overcome some compatibility setbacks which enabled me to try different parameters much more quickly.

Total Reward Per Episode



Average Predicted Q