# Playing Mario

Cody Kesler

March 24, 2020

## 1 Introduction

This work takes on the task of training a neural network to beat the first level of Super Mario Brothers, using deep reinforcement learning. Many 2-D and 3-D video games have been trained and beaten using applications of deep reinforcement learning. The current state of the art for reinforcement learning is given by Mnih, V., Kavukcuoglu, K., Silver, D. et al. in [1] as a DQN (deep Q-network) which learns successful policies directly from high-dimensional sensory inputs using end-to-end reinforcement learning.

## 2 Problem

The problem for this work is learning how to develop an action policy which can successfully complete World 1 Level 1 of Super Mario. The action space for the network is limited to [['NOOP'], ['right'], ['right', 'A'], ['right', 'B'], ['right', 'A', 'B'], ['A'], ['left']].

## 3 Data

The data required is given through an API built into the python GYM library known as gym-super-mario-bros. The library provides the state, actions, and rewards of the game to feed into the network.

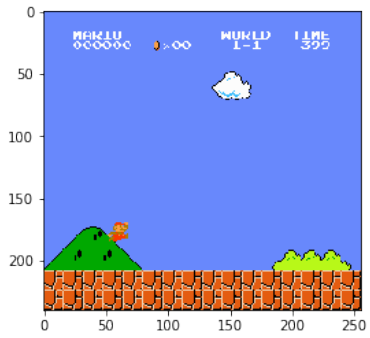


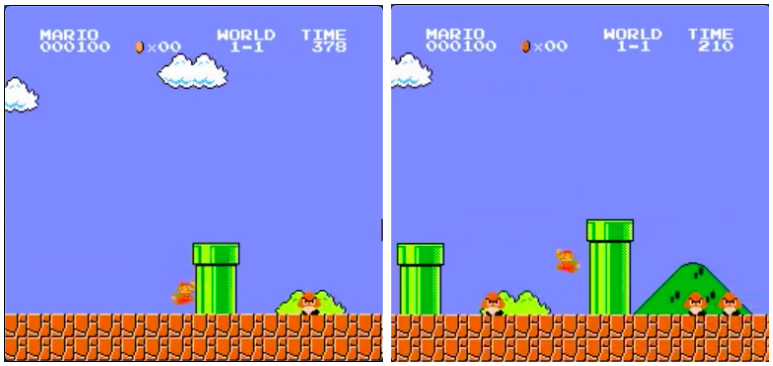

Figure 1: Image of Mario Gym Environment

## 4 Approach

Deep Reinforcement Learning has been well established for learning policies to play games from screenshots. By making a given number of rollouts, or game plays played by the initialized policy network, the reward is calculated with a Q learning algorithm. A value network is then trained to learn to calculate a reward from a given rollout. A policy network is then trained on the ratio of a new action to the old action multiplied by the reward for the given rollout. The policy network learns to maximize the reward multiplied by ratio of the two actions enabling it to improve its policy over many rollouts. Thus there was not a lot of experimenting when it comes to the general method of solving a reinforcement learning problem.

There was however some experimenting with the networks used and the set up of problem. The policy network, talked about in the next section needed to output probabilities from a convolution. Size of the image was important because of the memory restrictions of google colab and the way the algorithm runs. Because the algorithm collects a number of roll outs and stores them for the value and policy network, it needs a lot of memory. Thus in order to run the algorithm and still have information rich images, a size of 32x32 was used and the number of roll outs per epoch is 7. Because of the small amount of roll outs per epoch, a large amount of epochs is required to train the network to any reasonable amount of playing the game.

For the first few training's of epochs the output video of the network playing the game showed that would be stuck at the first and second tall pipe as seen in Figure 2.




(a) First Short Pipe (b) Second Tall Pipe

Figure 2: Stuck at Tall Pipes

We do note however that after a little longer training, Mario figured out how to jump over the first two pipes as seen in Figure 3. But still gets stuck at the 3rd tall pipe.


(a) Getting over First Short Pipe (b) Getting over Second Taller Pipe (c) Third Tall Pipe

Figure 3: Getting over Pipes

The issue of getting stuck at the pipes comes from the roll outs not having much chance to reach further in the level to experience getting over the pipes. This indicates a necessity for longer training allowing the network to have time to see what a tall pipe looks like and how to get over it. As seen from the ability for Mario to get over the tall pipes given some more training time, we train it for a larger amount of time and roll outs. The final results reachable by the given time constraints are shown below in the results section.

## 5 Convolutional Networks

Networks for both the Policy and Value learning needed to process screenshots from the game. The screen shots returned from the Super Mario Brothers gym environment is 255x245. They are resized to 64x64 with a cubic interpolation. The networks then take the image and process it through 3 convolutional layers and a fully connected layer. The number of channels through the network is $3 > 16 > 32 > 16$. The output is then reshaped into a 2304xn vector, where n is the number of actions in the action space.

Since the value network is only outputting 1 number, n = 1, and for the policy network which needs to output an action to take n = 7. For the policy network this vector is then passed through a softmax to get probabilities for the action to take.

## 6 Results

Unfortunately, more training of the networks was unable to produce better results as the furthest achieved was getting to the 3rd tall pipe. It was unable to get past the 3rd pipe with the time and power constraints.


Figure 4: Final Results for the farthest Mario reached on the First Level

## 7 Future Work

Given more resources, time, possibly a deeper network architecture and hyper parameter optimization the goal of getting to the end of the level will be achieveable. The next goal would be to finished all of the other levels of Super Mario Brother and ultimately beat the game.