Optimizing an Agent to Play Flappy Bird Using Deep Reinforcement Learning

Abstract
Simple video games, such as *Flappy Bird*, provide a good environment to measure the effectiveness of different neural network architectures. There is a need to determine the best neural network architectures for different systems so that more problems can be effectively solved using deep reinforcement learning. This work implements different variations of convolutional neural networks to learn to play the game *Flappy Bird.* The architectures tested in this work are based on previous works deep reinforcement learning and other machine learning projects. These studies showed a convolutional neural network based on the image classification architecture AlexNet is effective at learning the game *Flappy Bird* with deep Q-learning.

Introduction

Flappy Bird is a video game where a bird is navigated through sets of oncoming pipes. Although the game is simple, it is known for being notoriously difficult. A deep reinforcement learning agent can be trained to play *Flappy Bird* using a neural network. This study seeks to answer which neural network architectures are the most effective in training an agent to play *Flappy Bird*.

Methods

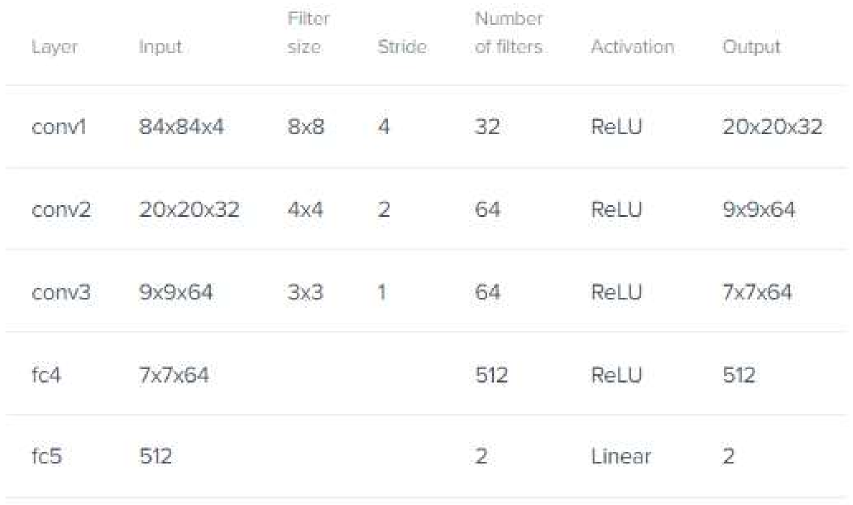
This project uses deep Q-learning to learn the game *Flappy Bird*. In Q-learning, a function computes the value of each state of the environment. When environments are more complex or have many states, a neural network can be used to replace this function. The addition of a neural network is the difference between Q-learning and deep Q-learning. There are many variations of neural networks and neural network architectures. Convolutional neural networks are often used to analyze images. In the case of *Flappy Bird*, the convolutional neural networks analyze frames of the game.

In order to properly compare how different neural network architectures perform, other aspects of each agent must be kept constant. Each agent played *Flappy Bird* for 425,000 iterations, making the decision to fly or fall each iteration. The agent is given a reward of 0.1 each iteration the bird remains alive, a penalty of -1 each time the bird dies. In order to incentivize the agent to fly through the pipes, the agent is given a reward of 10 every time the bird passes through a pipe.

The first agent tested is a baseline model. The agent chooses a random action every single iteration. There are two available actions to be randomly chosen, flying and doing nothing. Due to the nature of randomly guessing actions, his is the only agent which does not use deep Q-learning. The purpose of this agent is to show how effectively the game can be played without the use of any machine learning.

After the baseline, the next agents tested all utilize different convolutional neural network architectures. The neural network architecture tested is described in the DeepMind paper *Human Level Control Through Deep Reinforcement Learning* in which an agent was trained to play various Atari 2600 games. This architecture was chosen and implemented by Neven Pičuljan to teach an agent to play *Flappy Bird.* Table 1 describes the architecture of this neural network.

Table 1: Architecture of neural network implemented by Neven Pičuljan



Layer	Input	Filter size	Stride	Number of filters	Activation	Output
conv1	84x84x4	8x8	4	32	ReLU	20x20x32
conv2	20x20x32	4x4	2	64	ReLU	9x9x64
conv3	9x9x64	3x3	1	64	ReLU	7x7x64
fc4	7x7x64			512	ReLU	512
fc5	512			2	Linear	2

The final neural network architecture tested is based on the image classification neural network AlexNet. AlexNet is an architecture known for its success with the ImageNet dataset. AlexNet was not created with reinforcement learning in mind, but this architecture was chosen to test if a neural network with success in one area is able to be adapted to be successful with a different type of task. The architecture is implemented as follows:

```
self.conv1 = nn.Conv2d(4, 32, kernel_size=11, stride=4, padding=0)
self.maxpool = nn.MaxPool2d(kernel_size=3, stride=2)
self.conv2  =  nn.Conv2d(in_channels=32,  out_channels=64,  kernel_size=5,
stride= 1, padding= 2)
self.conv3  =  nn.Conv2d(in_channels=64,  out_channels=64,  kernel_size=3,
stride= 1, padding= 1)
self.conv4  =  nn.Conv2d(in_channels=64,  out_channels=64,  kernel_size=3,
stride=1, padding=1)
self.conv5  =  nn.Conv2d(in_channels=64,  out_channels=64,  kernel_size=3,
stride=1, padding=1)
self.fc1  = nn.Linear(in_features=5184, out_features=512)
self.fc2  = nn.Linear(in_features=512, out_features=512)
self.fc3                  =                  nn.Linear(in_features=512,
out_features=self.number_of_actions)
```

Results

As each model is trained, the average reward for the previous 25,000 iteration is plotted. The larger the average reward, the better the model has learned how to play *Flappy Bird.* For the sake of this experiment, a model which can reach a higher reward in a lower number of iterations is considered a more successful model.

The first agent to give a result is the random agent. Over 425,000 iterations, the average reward remained constant. Because it is unlikely to randomly get the correct series of inputs, the bird runs into the first pipe almost every single time. Figure 1 shows the average reward every 25,000 iterations over 4525,000 iterations.

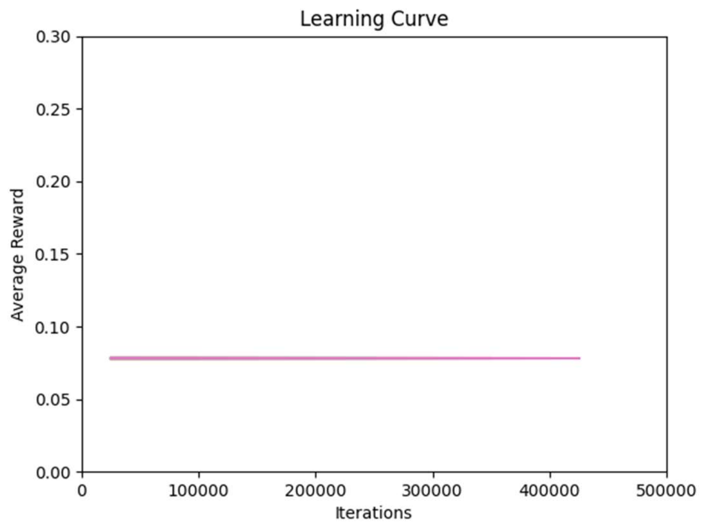


Figure 1: Average reward when taking random actions

The neural network architecture from the DeepMind paper *Human Level Control Through Deep Reinforcement Learning* is used by the next agent. This agent showed the ability to learn, and the average reward began to steadily increase near iteration 350,000. At iteration 425,000 the average reward was estimated to be 0.14. Figure 3 shows the learning curve for this agent.

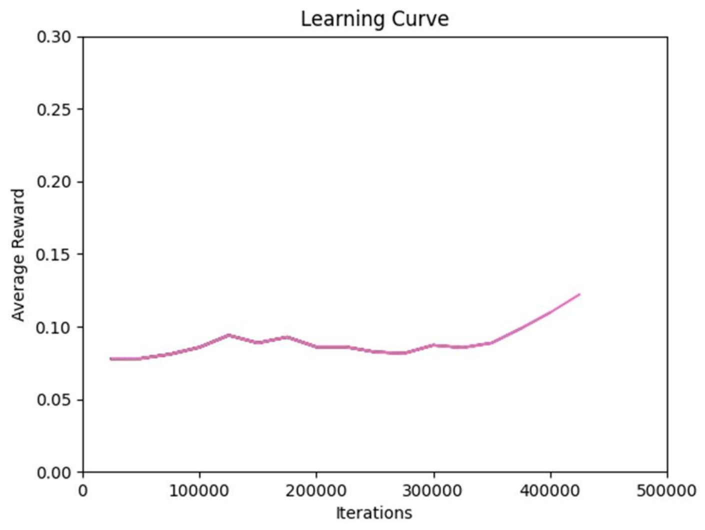


Figure 3: Average reward using architecture from *Human Level Control Through Deep Reinforcement Learning*

The final agent tested used a neural network architecture based on AlexNet. The results for this architecture showed the ability to learn. The average reward with this architecture rose above 0.10 after an estimated 50,000 iterations.

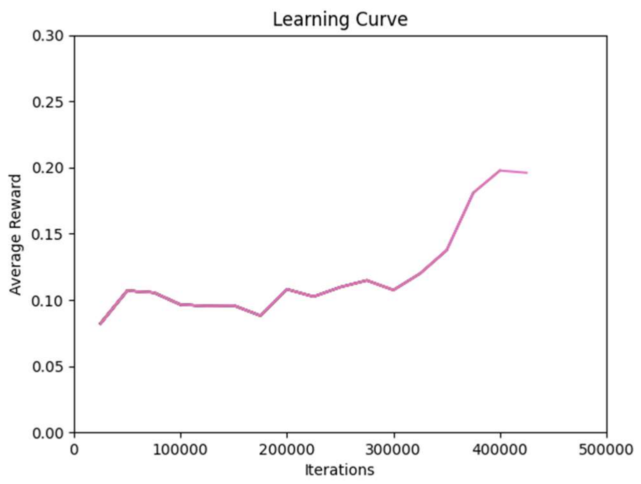


Figure 3: Average reward using architecture based on AlexNet

Comparing the average reward of all four models deep Q-learning models are able to learn the game *Flappy Bird.* The random action model shows an agent without the ability to play the game, meaning any model above this baseline has at least partially learned how to play. Of the models tested, the AlexNet architecture performed the best. In the 425,000 iterations, the AlexNet Architecture reached an average reward of 0.20, and the DeepMind architecture reached 0.14. Based on these results, convolutional neural networks known to perform well with other tasks will perform well with deep reinforcement learning tasks.

Summary

Flappy Bird is a game where the player navigates a bird through sets of oncoming pipes. The only possible actions the player can take are to fly up or to do nothing and fall. Four models are implemented to learn the game *Flappy Bird*. These models include a baseline where a random action is always chosen, a deep Q-learning model using a convolutional neural network defined in a DeepMind research paper, and a model using a convolutional neural network based on AlexNet. After running all models for 425,000 iterations, the AlexNet architecture provided the best results in the smallest number of iterations.

Conclusion

The AlexNet convolutional neural network architecture performed the best of all agents trained to play *Flappy Bird* in this project. Testing multiple agents showed which types of agents are able to solve reinforcement learning problems efficiently. With more time, more neural network architectures could be tested to better discover which attributes of a network are correlated with successfully learning reinforcement learning problems.

Sources

Code for this project: https://github.com/jakewpope/DeepRLFlappyBird

Based on the code written for this tutorial: https://www.toptal.com/deep-learning/pytorch-reinforcement-learning-tutorial

4 CNN Networks Every Machine Learning Engineer Should Know.
https://www.topbots.com/important-cnn-architectures/

Human Level Control Through Deep Reinforcement Learning.
https://deepmind.com/research/publications/2019/human-level-control-through-deep-reinforcement-learning