Introduction to Reinforcement Learning

Project 1: Navigation Report

Jose Lara | jose.lara.l@outlook.com

Problem Statement

The goal of this project is to train an agent to navigate a Unity environment to maximize rewards. To maximize reward, the agent must collect as many yellow bananas as possible while avoiding blue bananas in Unity's Banana environment.

Using a Deep Q Network, based on Pytorch library, and Deep Q Learning algorithm, a final set of optimized weights must generate an optimal solution. The RL agent based of these weights must be able to average a reward of at least 13, over 100 episodes.

Design

Deep Q Network

The Deep Q Network consists of two hidden layers, each with 64 nodes. A rectified linear unit function is applied between each layer.

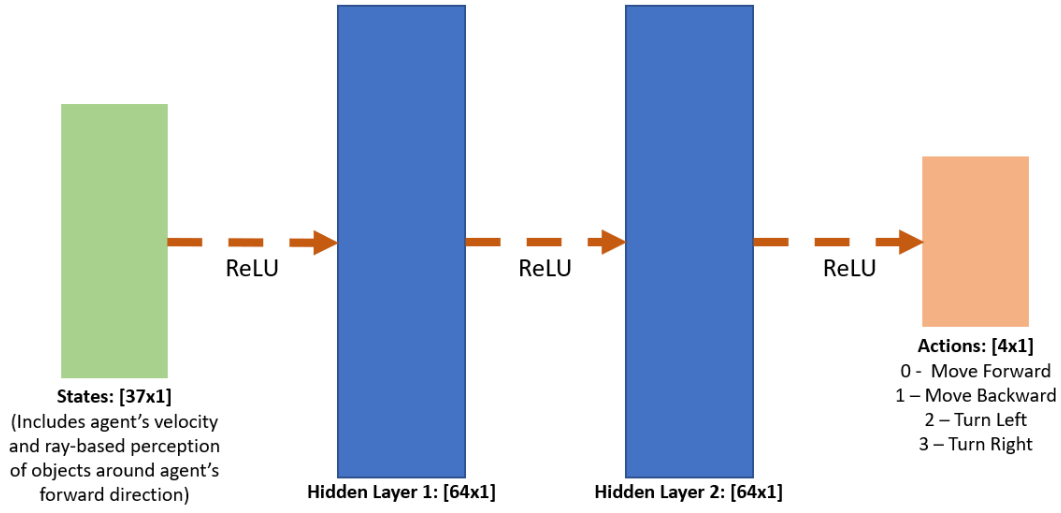

Figure 1: Deep Q Network configuration.

Algorithm

The algorithm used mainly stems from the example provided in the Deep Q Learning exercise in the **Lesson 2: Deep Q-Networks**. The Double Deep Q Learning was extension was implemented to improve the score after failing to get an average reward of +13.

Training Results

The initial algorithm used the same hyperparameters used in the original Deep Q-Network assignment. However, the training did not return an average reward of +13 at any point during the training.

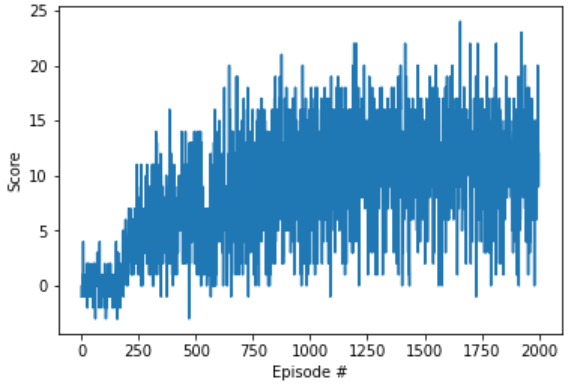

Figure 2: Standard Deep Q-Learning training did not reach an average +13

Since the original Deep Q-Network was not meeting the goal, I implemented the Double Deep Q-Learning (Double DQN) Algorithm. Though after further troubleshooting the training, I found that the reason my training wasn't working was due to not adding the rectified linear activation function between each layer in the Deep Q Network.

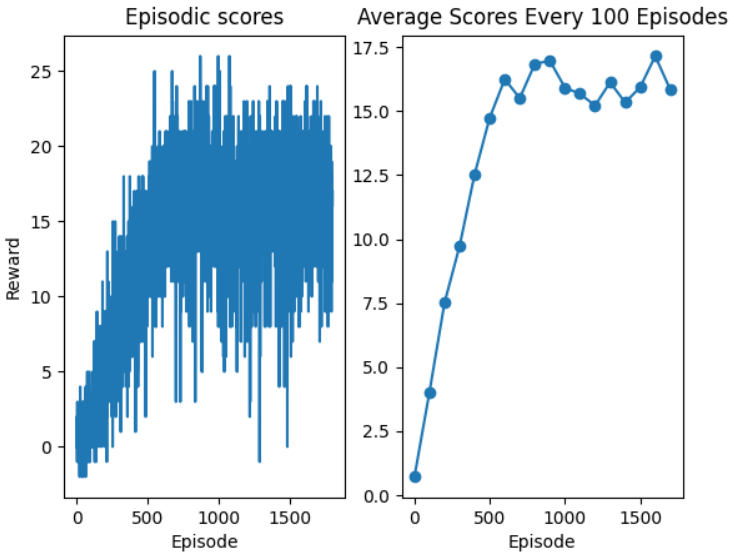

Figure 3: Training results after updating Deep Q Network with ReLU and implementing Double DQN.

To test the results, a new agent was loaded using the optimized weights and tested for 100 episodes. Below are the results with the average value labeled.

Figure 4: Earned rewards using trained agent, with an average of 15.74 in rewards after 100 episodes.

Next Steps

First steps I want to take with this project is generalizing the code to work with multiple Udacity environments. Creating a list of all applicable environment, and using the correct state values, build an applicable Deep Q Network.

Next, I would like to build an optimization script that iterates through hyperparameters at the model and algorithm label. These parameters include number of nodes in each hidden layer, buffer size and learning rate.

Furthermore, I would like to implement the five remaining Deep Q-Network algorithm extensions that address different limitations in the original learning approach. Implementing the Rainbow algorithm should maximize the reward score while training, as shown by the results of Google DeepMind's work training agents to learn how to play 57 Atari games.

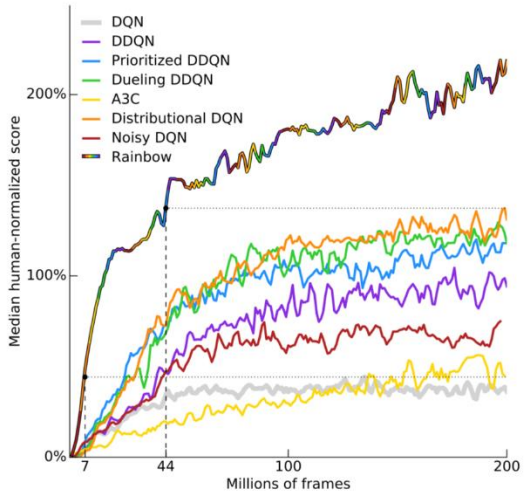

Figure 6: Example of Rainbow algorithm implementation used to learn 57 Atari games by Google DeepMind.