Report

1. Introduction

This is a report file that details the steps done by us to solve the "Bananas" Environment. It describes the learning algorithm used, along with the number of steps performed to solve the task and the extensions made by our team that we plan to do in the future.

2. Learning Algorithm

As described in the README, the algorithm is based on ths Udacity's Deep Reinforcement Learning Nanodegree exercise on Deep Q Networks. Just like in that file, we divided the task into 3 files, which are described in the README. The modifications from that implementation are as follows:

- Model.py : just like in the DQN workspace, we use a three layer ANN, with the first layer consisting of sixty-four hidden units, just like the second layer. The number of units of the final layer is equivalent to the number of actions spaces in the environment, in this case four. We used RELU activation units throughout the ANN.

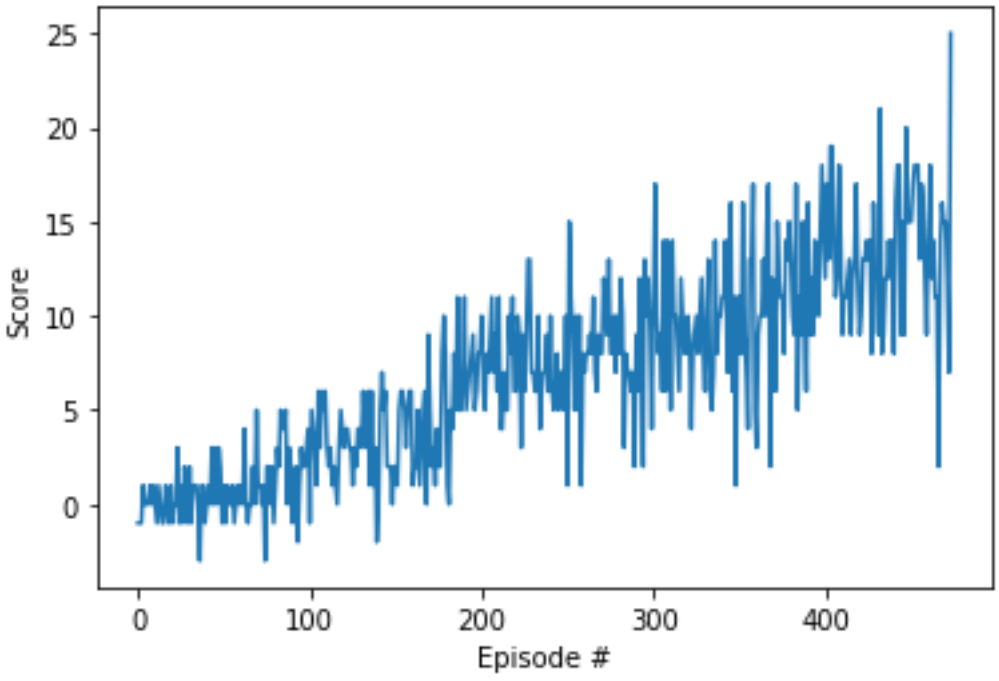
- dqn_agent.py: The implementation of the DQN agent as well as the buffer is very similar to the one in the previous task, the main difference, as stated by our README, consists of the fact that the agent doesn't directly control the environment but acts through a so called "brain".

- Navigation.ipynb: The flow of the notebook is similar to the one in the previous task, but differs in several reasons: the dependencies are different since we are using a Unity environment, whereas we used OpenAI gym for the previous assignment. The code to make the agent act differs by the "brain" operator, among other consideration.

3. Results

The agent was able to achieve the desired score over 374 episodes, which is way below the mark hinted by the instructors

Training graph summary



4. Considerations for future work

We believe there are four ways we are looking to improve the algorithm. First of all, we consider the ANN to be of a rather simplistic nature, and we are looking forward to implement a more complex architecture to see the model's performance. Second of all, we're looking forward to accept the challenge and implement a solution to this environment based solely on a CNN that looks at the pixels provided by the environment. Third, we're currently trying our best to implement improvements to the DQN model, namely Double DQN and Dueling DQN. Finally, we're excited to make it all come together into a beautiful Rainbow algorithm implementation that will knock your socks off. Please stay tuned the developments in this repository in the coming weeks.