Project 1: Navigation

Technical Project Report

1. Description of the Implementation

The Jupyter notebook includes the following sections:

1. **Start the Environment:** Installing the packages in Jupyter and setting up the banana environment in Unity,
2. **Examine the State and Action Spaces:** Using print statements to learn about the environment,
3. **Take Random Actions in the Environment:** this is both randomly exploring the environment and the Deep Q algorithm implementation

2. Learning Algorithm

The algorithm used is a Deep Q learning algorithm. The DQN architecture is composed of a couple of convolution layers, followed by a couple of fully connected layers. At a high level this algorithm works by taking images from the banana navigation game and producing a vector of actions. The algorithm is then fed back the number of bananas collected. At the beginning, the algorithm explores randomly but over time became better.

Hyperparameters used are:

Hyperparameter	Description	Value
n_episodes (int)	maximum number of training episodes	2000
max_t (int)	maximum number of timesteps per episode	1000
eps_start (float)	starting value of epsilon, for epsilon-greedy	1.0
eps_end (float)	minimum value of epsilon	0.01
eps_decay (float)	multiplicative factor (per episode) for decreasing epsilon	0.995

Table 1: Description of Hyperparameters

3. Plot of rewards

The environment was considered solved when the agent was able to receive an average reward (over 100 episodes) of at least +13. The agent needed 379 episodes to solve the environment.

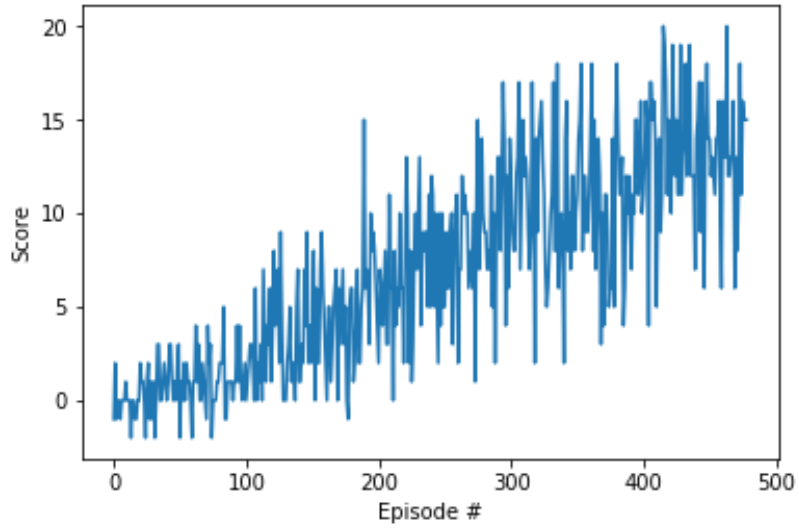

Figure 1: Plot of rewards

4. Ideas for future work

This algorithm could be further improved by implementing the following methods Double DQN, Prioritised Experience Replay and Duelling DQN.

4.1. Double DQN

This method improves on the problem of overestimating the Q-values resulting in faster and more stable learning. Overestimation occurs when exploration from a state is low which means there is high uncertainty on which is the best next action to take.

4.2. Prioritised Experience Replay (PER)

This method aims to increase the agent's exposure to of less frequent and high importance experiences. PER looks to improve on DQN's random sampling by prioritising samples from experiences randomly.

4.3. Duelling DQN (DDQN):

This method aims to accelerate training by limiting the number of times Q(s,a) is calculated. Only the value of being in that state is calculated and not the effect of taking further actions. This is particularly useful when there are a lot of actions that can be taken that are irrelevant to the agent objective.