Project 1: Navigation

Environment Details

For this project, we will train an agent to navigate a large, square world and collect as many yellow bananas while avoiding the blue bananas.

A reward of +1 is provided for collecting a yellow banana, and a reward of -1 is provided for collecting a blue banana. Thus, the goal of our agent is to collect as many yellow bananas as possible while avoiding blue bananas.

The state space has 37 dimensions and contains the agent's velocity, along with ray-based perception of objects around the agent's forward direction. Given this information, the agent has to learn how to best select actions. Four discrete actions are available, corresponding to:

- 0 - move forward.
- 1 - move backward.
- 2 - turn left.
- 3 - turn right.

The task is episodic, and in order to solve the environment, your agent must get an average score of +13 over 100 consecutive episodes.

Learning algorithm

We solve the problem here with the Double DQN algorithm. The algorithm estimates the value function for a state action pair and chooses action based on this.

NETWORK ARCHITECTURE - Fully connected layer with three hidden layers.
Input size - state_size(37)
Output size - action_size(4)

The first hidden layer has 256 nodes, the second has 128 nodes and the third has 32 nodes.
The activation function used here is the relu activation function.

The hyperparameters for the problem are as follows,

```
BUFFER_SIZE = int(1e5)  # replay buffer size
BATCH_SIZE = 64         # minibatch size
```

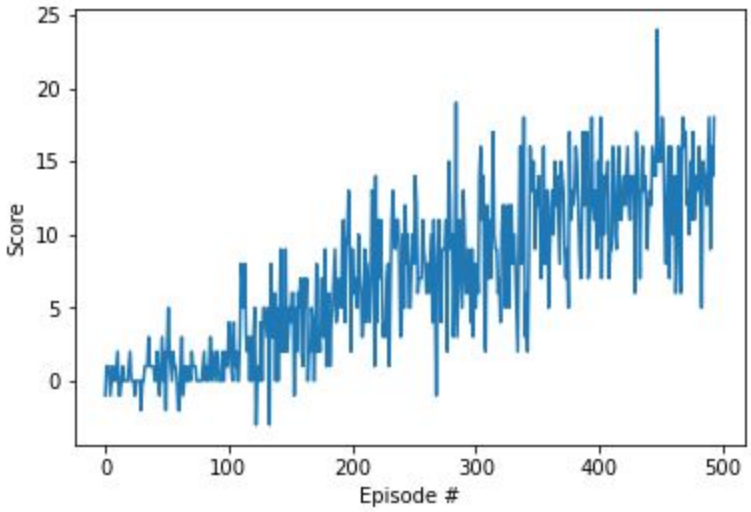
```
GAMMA = 0.99           # discount factor
TAU = 1e-3             # for soft update of target parameters
LR = 5e-4              # learning rate
UPDATE_EVERY = 4       # how often to update the network
SAMPLE_DELTA = 0.05
```

We have taken a batch size of 64 as we have enough data and it is not too high that it would slow down learning.

We have taken a gamma as 0.99 since we care about current reward as well as future reward.

The learning rate is kept low so we are able to maintain stability and ensure convergence.

Below is the plot of the score vs episode number for the trained agent. We solve the problem in 394 episodes.



Future Work

I would like to add prioritized experience sampling as well. I have the code for it but current implementation seems too slow. Will try to use the Sum Tree data structure to improve on it.

I would also like to implement the Duelling DQN to compare it against DQN and Double DQN on how well it learns in this environment.

In terms of improvement, I noticed that there is a lot of variance during our training and naturally in the performance of our agent. I would like to try implementing the Averaged-DQN (https://arxiv.org/pdf/1611.01929.pdf) to see if this can help reduce the variance.