Project 1 : Navigation

Problem Statement

We have to train an agent to navigate (and collect bananas!) in a large, square world.

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 your agent is to collect as many yellow bananas as possible while avoiding blue bananas.

Environment Analysis

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. To win, the target was set to achieve an average score of +13 over 100 consecutive episodes.

Implementation Details

The approach followed was:

- Understand the state and action space
- Impementation of Deep Q Learning Model
- Train the agent with the above model over different epochs till the target was achieved
- Plot the results of learning across episodes

Algorithm and Implementation

The algorithm used was Deep Q Learning algorithm (https://storage.googleapis.com/deepmind-media/dqn/DQNNaturePaper.pdf) that uses Deep Neural Networks with taking agent state as input and Q action values as output. It also uses Experience Replay and Target Network – both of which help stabilizing model training.

Architecture:

- The Q Network used is a fully connected Neural Network with 3 layers:

First layer: 64 neurons, Second layer: 64 Neurons, Third layer: Size of Action Space (4). All input connected layers were relu actived.

- The DQN used Epsilon Greedy Action Algorithm
- The DQN's aim was to approximate Q-Function to find the optimial policy that maximized reward of the agent
- Experience Replay was used to allow agent to learn from previous experiences. The replay used a buffer and fetched experiences at random.

Hyperparameters Used for learning:

Replay buffer size = 1e5

Mini Batch Size = 64

Discount factor (Gamma) = 0.99

Soft update for Target Parameters (Tau) = 1e-3

Learning Rate (For Adam Optimizer) = 5e-4

Update Interval = 4

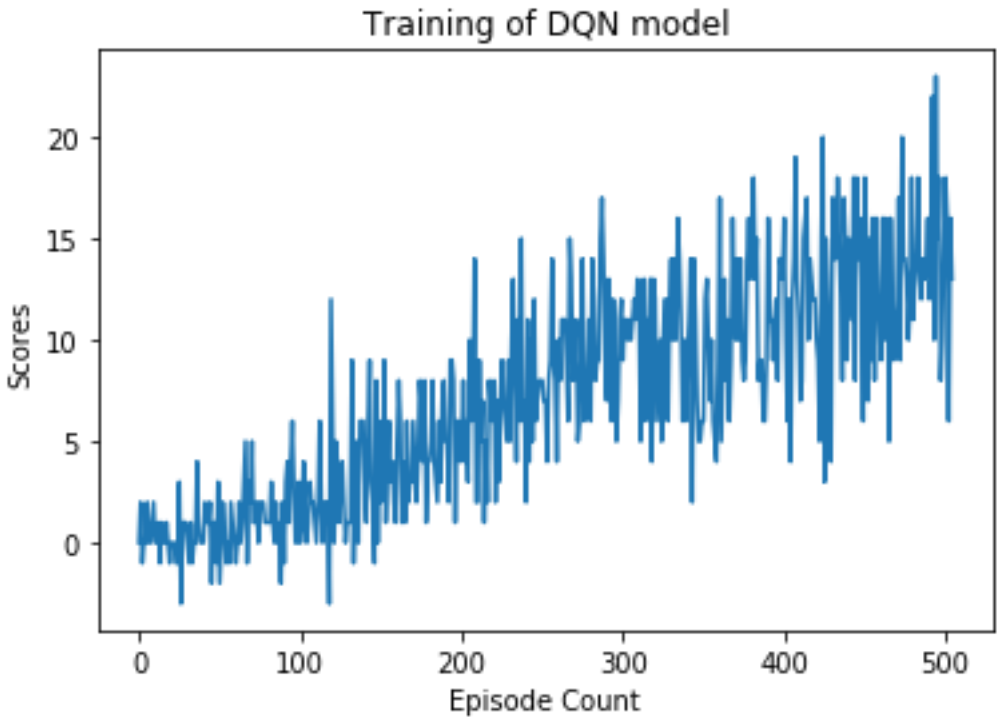
Epsilon Start (for eps greedy action selection) = 1.0

Epsilon Decay = 0.995

Epsilon End = 0.05

Results

The agent was able to navigate the environment in 505 Episodes (which is well below 1800 as mentioned in the project rubric here: https://review.udacity.com/#!/rubrics/1889/view)



Output plot of scores vs episode count

Enhancements for the future

- Add more layers to neural network to improve the efficiency of the training
- Tweak hyperparameters or use hyperparameter search to ensure learning is achieved quickly with better performance
- Use Double DQN, Dueling DQN or Prioritized Experience Relay over DQN and evaluate performance, all of which help in faster learning
- Try using pixel based navigation and compare the output