Project Report

This project utilised the double deep Q-netword (DDQN) architecture train the agent to navigate in a world full of bananas.

State and Action Space

The simulation contains a single agent that navigates a large environment. At each time step, it has four actions at its disposal:

- 0 - walk forward
- 1 - walk backward
- 2 - turn left
- 3 - turn right

The state space has 37 dimensions and contains the agent's velocity, along with ray-based perception of objects around agent's forward direction.

Learning algorithm

The learning algorithm is implemented in the Agent.learn() method. There is a switch in the Agent class constructor to determine if we are going to use double of vanilla dqn to train the agent.

Hyperparameters

BUFFER_SIZE: The size of replay buffer, set to 100000 entries

BATCH_SIZE: The size of mini batch, set to 64

GAMMA: The discount factor, in this case there is no difference getting yellow banana earler or later, so it is set to 1

TAU: The factor for soft update of target parameter. Double dqn does not work well when the TAU is small. So in my case, I set TAU = 1 when using double dqn. If we choose vanilla dqn, TAU can be set to 0.001 to get satisfying result. I think the soft update has similar idea of double dqn, which is decouple the correlation between local network and target network as much as possible. A small TAU makes target network follows local network with big delay.

eps_start: The starting value of epsilon, for epsilon-greedy action selection. It is set to 0.3.

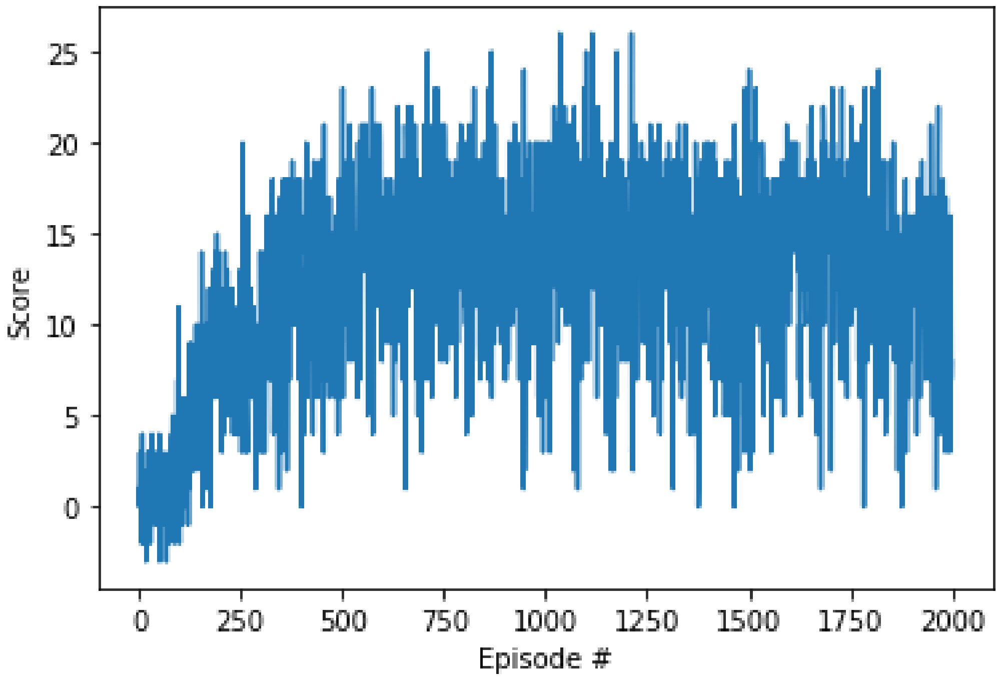
eps_end: The minimum value of epsilon. It is set to 0.01

eps_decay: The multiplicative factor (per episode) for decreasing epsilon. It is set to 0.995.

Neural Network

The Q-network utilise 2 x 64 Fully Connected Layers with Relu activation followed by a final Fully Connected layer with the 4 actions as the action size. The network has an input dimension of 37, which is same as the size of state space.

Plot of Rewards



```
Episode 100	Average Score: 1.80
Episode 200	Average Score: 5.32
Episode 300	Average Score: 8.55
Episode 400	Average Score: 11.66
Episode 500	Average Score: 13.80
Episode 600	Average Score: 13.66
Episode 700	Average Score: 15.06
Episode 800	Average Score: 15.47
Episode 900	Average Score: 15.21
Episode 1000	Average Score: 14.94
Episode 1100	Average Score: 14.64
Episode 1200	Average Score: 14.76
Episode 1300	Average Score: 12.66
Episode 1400	Average Score: 14.86
Episode 1500	Average Score: 15.17
Episode 1600	Average Score: 13.65
Episode 1700	Average Score: 15.44
Episode 1800	Average Score: 13.64
Episode 1900	Average Score: 12.16
Episode 2000	Average Score: 13.07
```

The environment is solved at between 300-400 episode.

Ideas for Future Work

There are a few things left:

1) Try different depth / width of the neural network to compare the performance;

2) Introduce the convolution neural network into the system to train the agent based on the pictures captured by the video camera instead of the 37 sensors.