# Banana Project Report

This project was solved using DQN (deep Q learning network). Regarding the neural network, two hidden layers were used. The first layer contains 128 neurons, and the second hidden layer is 64 neurons.  The output layer has a size of 4 as per action size.

The main parameters used during the learning were:

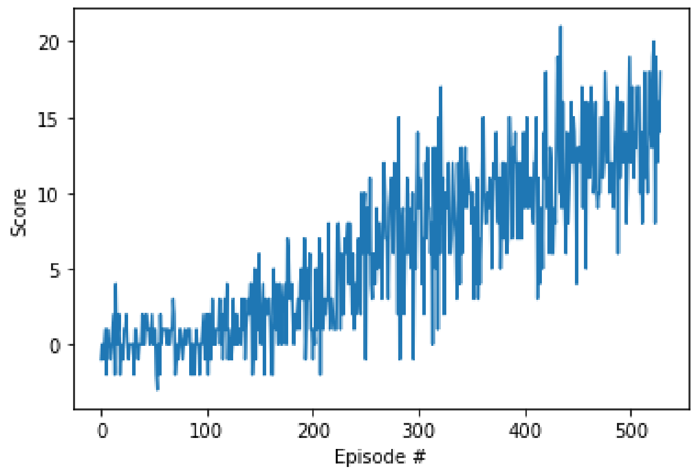
- Number of episode (=1000) I expected to converge before this number
- Time per episode (=1000) considered enough for training
- Epsilon start value = 1
- Epsilon end value = 0.01
- Epsilon decay = 0.995
- Replay buffer size = 100,000
- Gamma = 0.99 to give higher priority for current reward
- Learning rate = 0.0005

The device used is the CPU.

The program of my_solution called the dqn_agent.py and created an agent based on that.

The result of the program came as follows:

```
Episode 100	Average Score: 0.21
Episode 200	Average Score: 1.77
Episode 300	Average Score: 5.50
Episode 400	Average Score: 9.08
Episode 500	Average Score: 11.77
Episode 529	Average Score: 13.08
Environment solved in 529 episodes!	Average Score: 13.08
```



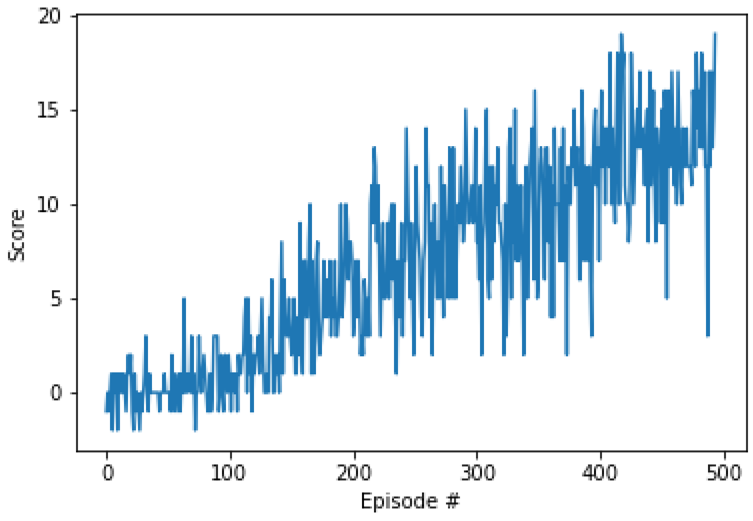
The above figure indicates that the agent was able to get an average score above 13 in 100 consecutive episodes starting from episode 430 till 529.

**Double DQN:**

The double Deep Q learning network was tried in the above-mentioned project, and it included the same parameters and shape of neural network. However, there was small change in the update of the target network in double_dqn_agent.py to avoid overestimating.

The result for the double DQN came as shown below:

```
Episode 100	Average Score: 0.35
Episode 200	Average Score: 3.55
Episode 300	Average Score: 7.56
Episode 400	Average Score: 9.57
Episode 494	Average Score: 13.07
Environment solved in 494 episodes!	Average Score: 13.07
```



The project was solved from episode 395 and 494 with an average score above 13. This is more than 5% improvement of the standard DQN method.

## Future Work:

For this project it is expected to give a better result if used Deul DQN along with other improvement methods (like Rainbow) The collection of the improvement method is supposed to converge faster.