Report – Navigation

Environment

The code was tested on workspace in udacity course, using the environment“Banana.x86_64”. This environment is trying to make the agent learn to catch yellow bananas as much as possible while avoiding blue bananas.

Algorithms

The main algorithm is based on Deep Q-learning considering “Experience replay” and “Fixed-Q-Targets”. Specifically, the policy model is a neural network, which has 37 states as input and 4 action as output. There are two linear hidden layers (64 neurons), using ReLU as activation function. The concrete architecture is shown below in Figure 1.

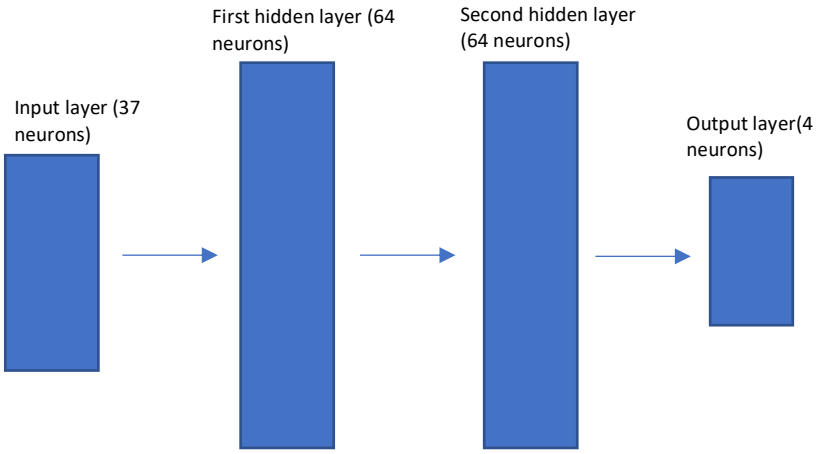


Figure 1

Next step is to train the agent. We use experience replay, namely storing (states, actions, rewards, next steps, done) into a Replay Buffer, to help the agent learn all possible situation. Since the weight of the Q-networks is the same as the weights of the goal, it’s difficult to converge. Therefore, a fixed Q-target is introduced, which trains local Q-Network with the same architecture and then soft-update the target Q-Network.

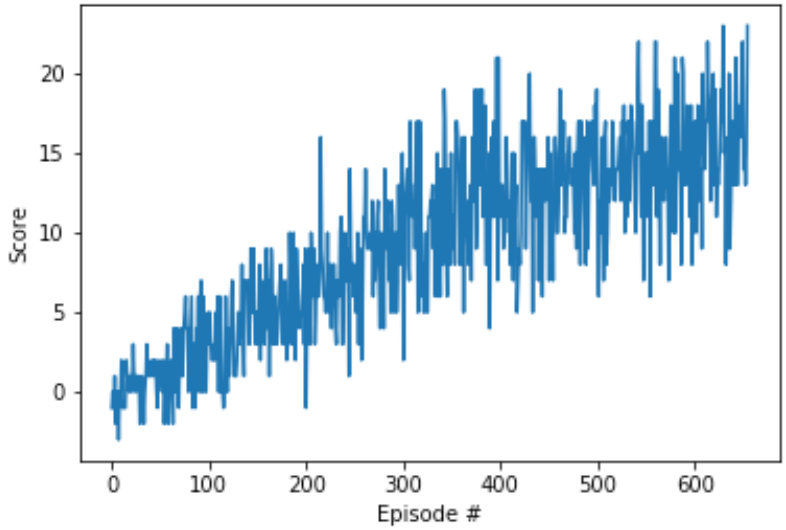
It’s worth to mention that an epsilon-greedy action policy(exploration and exploitation) and a discount value, gamma(0.99) are introduced during learning.

Results.

With given algorithm, the average score can achieve 15 after 655 episode. See the graphs below.

```
Episode 100     Average Score: 1.26
Episode 200     Average Score: 4.60
Episode 300     Average Score: 7.77
Episode 400     Average Score: 11.66
Episode 500     Average Score: 12.89
Episode 600     Average Score: 14.12
Episode 655     Average Score: 15.14
Environment solved in 555 episodes!     Average Score: 15.14
```

```
: import matplotlib.pyplot as plt
```



Future work

There are multiple improvements could be done: eg. Double DQN to improve the robust of the model and Prioritized Experience replay so as to help the agent learn more from the important samples. In addition, a test environment may also help to test the model.