Udacity Deep Reinforcement Learning

Project 1: Navigation

Jihye Seo

Feb. 11,2020

Introduction

In this exercise, I have trained an agent that collects banana with various version of Deep Q-Network(DQN). The agent receives a reward of +1 for collecting Yellow banana, while it receives -1 reward for collecting Blue. The goal was to gain reward of +13 over consecutive episodes. I have observed that the goal was achieved around 300~ 400 episodes.

I have tested different combination of DQN algorithms: vanilla DQN, Double DQN, Dueling DQN. The one that utilized both Double and Dueling techniques gave the best results in terms of maximum score achieved and the speed of the training.

Deep Q-Network Architectures

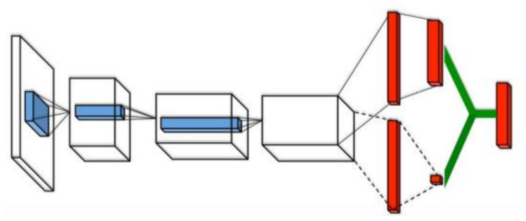
Our agent has 37 state and 4 action spaces. Sampled states from replay buffer were fed into the Deep Neural network as inputs. I have tested out two type of neural network. First network had 2 hidden layers of 64/32 nodes each with relu activation while second network had 3 hidden layers of 128/64/32 nodes each also with relu activation.

Deep Q-Learning methods

I have experimented different type of DQN. Double DQN uses local Q network to pick next argmax(Q(s,a)) action and uses target network to compute target Q values to reduce overestimation of Q-Values.

$$Y_t^{\text{DoubleQ}} = R_{t+1} + \gamma Q(S_{t+1}, \underset{a}{\operatorname{argmax}}\, Q(S_{t+1}, a; \boldsymbol{\theta}_t); \boldsymbol{\theta}'_t)$$

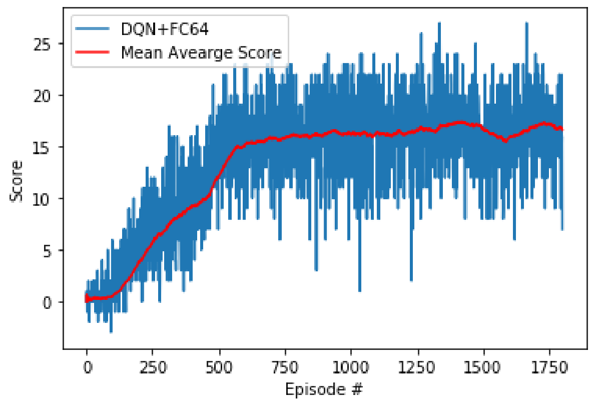
Duel DQN was used to determine which states are (or are not) valuable. I have changed the Deep Q-network architecture so that v(s) can be combined and used for assessment of each state.

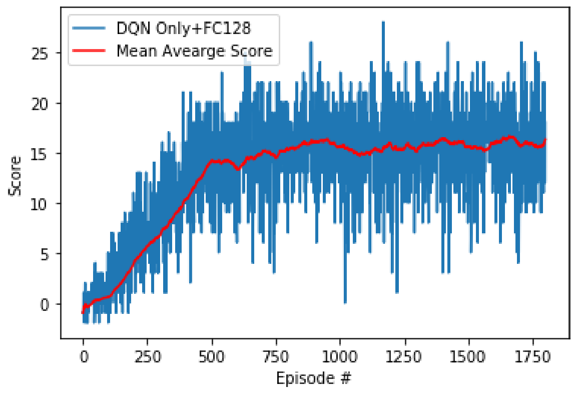


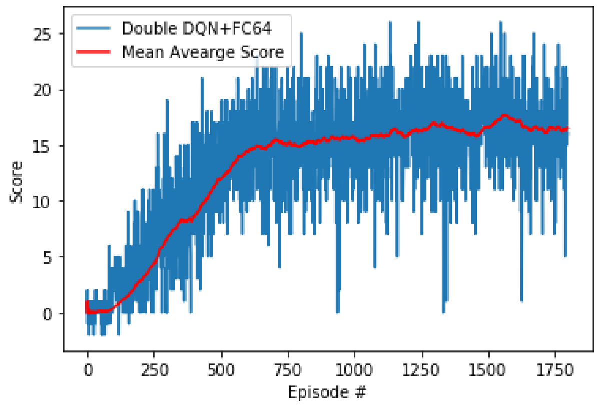
Results

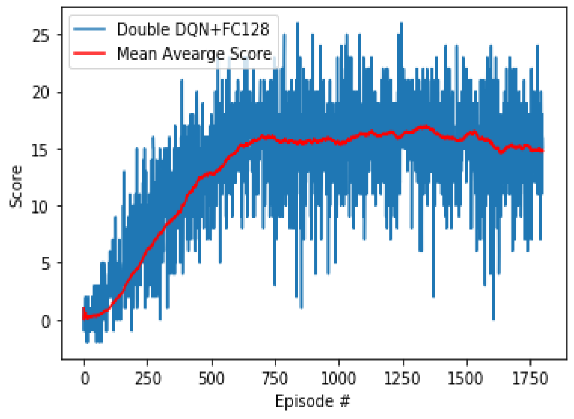
The results for different combination of Q network architecture and DQN algorithms are shown below. Neural network that of 2 layers seemed to work better than 3 layers in terms of both rewards and speed. The one that utilize either double DQN or Double+Deuling DQN showed better rewards, in terms of Max mean Scores, but in term of speed it gave us confusing results. We may have to try multiple trials to get the average effects of each algorithms.

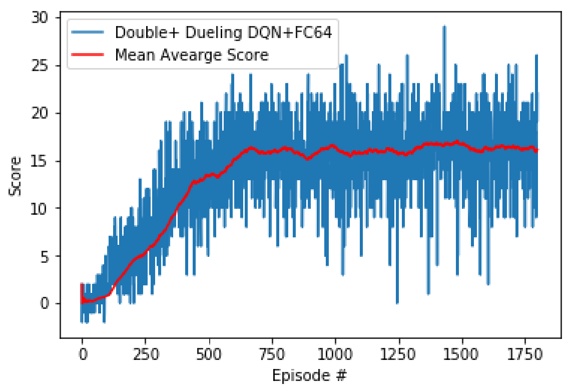
```
  [duel,double,network] MaxScore MaxMeanScore F_Episode
0            [0, 0, FC64]       27        17.43       419
1            [0, 1, FC64]       26        17.71       439
2            [1, 1, FC64]       29        17.02       368
3           [0, 0, FC128]       28        16.64       372
4           [0, 1, FC128]       26        17.02       421
5           [1, 1, FC128]       26        16.79       413
```

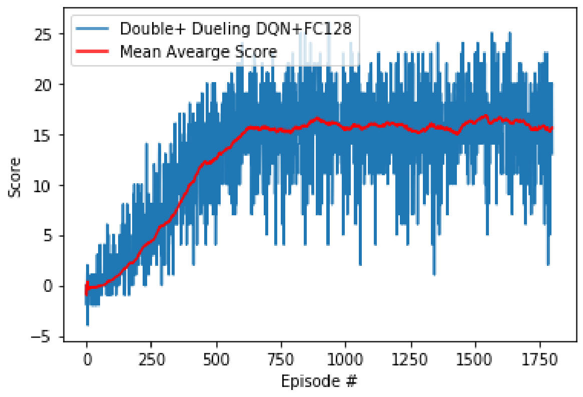












Future Work

While I experimented different neural networks and DQN algorithms there are still many ways of improvement of my model.

1. Hyperparameter tuning

There are many hyper parameter options such as Batch size, soft update Value, Learning rate, Update Frequency, Discount Factor etc. to tweak, but I think ε would give different significant differences in Reinforcement Learning. I used default epsilon values from previous exercises. We can adjust ε to control degree of exploration. By increasing ε decay , I may able to decrease the randomness of the agent action and learn fast since it seems that our environment does not have many states to explore.

$$\varepsilon = \max(\varepsilon_{end}, \varepsilon * \varepsilon_{\text{decay}}).$$

$$\varepsilon_{\text{start}}\text{=1.0, } \varepsilon_{end}\text{=0.01, } \varepsilon_{\text{decay}} = 0.995$$

2. Deep Q Network tuning

I tried two different neural network. But simpler one with 2 hidden layers gave me better results. Regarding the number of nodes to be trained, simpler networks may converge faster but the results of maximum rewards can be different. Different number of hidden layers or number of nodes in each layer can be experimented for the future work.

3. Utilizing Prioritized sampling DQN

This will allow the agent run more significant experiences more often. I think this would give faster convergence and rewards.