DRLND Project 1 Report

Introduction

In this project, I trained an agent to navigate (and collect bananas) in an environment provided by Unity-ML.

There will be two types of bananas, yellow and blue. The agent is trained to collect ONLY yellow bananas and avoid blue ones. The agent receives +1 reward for collecting yellow bananas and -1 for collecting blue bananas.

The environment has 37 dimensions state space and there are 4 discrete actions available: move forward (0), move backward (1), turn left (2), turn right (3).

The aim of this project is to get an average score of +13 over 100 consecutive episodes.

Learning Algorithm

I used deep neural networks to approximate the reinforcement learning components: value function, policy, and model (state transition and reward function).[1]

I searched for similar works as this project, and I found that 2 fully connected layers was solved in more episodes compared to 3 fully connected layers.[2,3]

In this project, I implemented 3 fully connected layers and an output layer with 4 action values).
I tried different fc layers and the results of those different layers were not much different.

1st experiment	2nd experiment	3rd experiment	4th experiment	5th experiment
fc1 : 128 fc2 : 64 fc3 : 32 Solved in episode 441	fc1 : 62 fc2 : 32 fc3 : 16 Solved in episode 403	fc1 : 256 fc2 : 128 fc3 : 64 Solved in episode 497	fc1 : 32 fc2 : 16 fc3 : 8 Solved in episode 604	fc1 : 512 fc2 : 256 fc3 : 128 Solved in episode 468

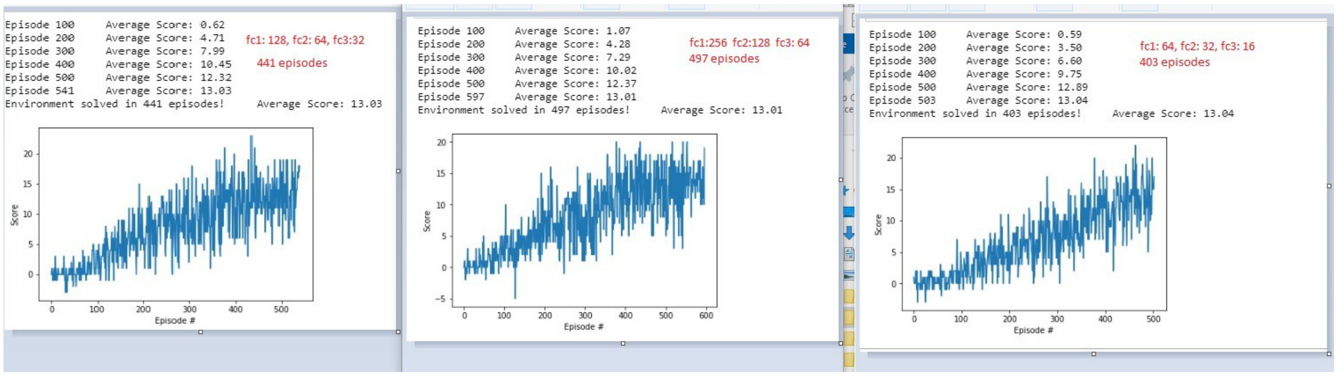
I submitted 2nd experiment for this project.

Hyperparameters:

- BUFFER_SIZE= 1e5
  The replay buffer size
- BATCH_SIZE= 64
  Number of inputs processed per batch when running Stochastic Gradient Descent
- GAMMA= 0.99
  Discount factor of the Q-Learning Algorithm
- TAU: 1e-3
  To perform soft updates of the target network parameters
- LR: 5e-4
  Learning rate provided to the Adam optimizer
- UPDATE_EVERY= 4
  It is used to determine how often to update the network

Plot of Rewards

Below is the rewards plot compilation from different experiments I did



```
Episode 100     Average Score: 0.62
Episode 200     Average Score: 4.71
Episode 300     Average Score: 7.99
Episode 400     Average Score: 10.45
Episode 500     Average Score: 12.32
Episode 541     Average Score: 13.03
Environment solved in 441 episodes!     Average Score: 13.03
```

fc1: 128, fc2: 64, fc3:32

441 episodes

```
Episode 100     Average Score: 1.07
Episode 200     Average Score: 4.28
Episode 300     Average Score: 7.29
Episode 400     Average Score: 10.02
Episode 500     Average Score: 12.37
Episode 597     Average Score: 13.01
Environment solved in 497 episodes!     Average Score: 13.01
```

fc1:256 fc2:128 fc3: 64

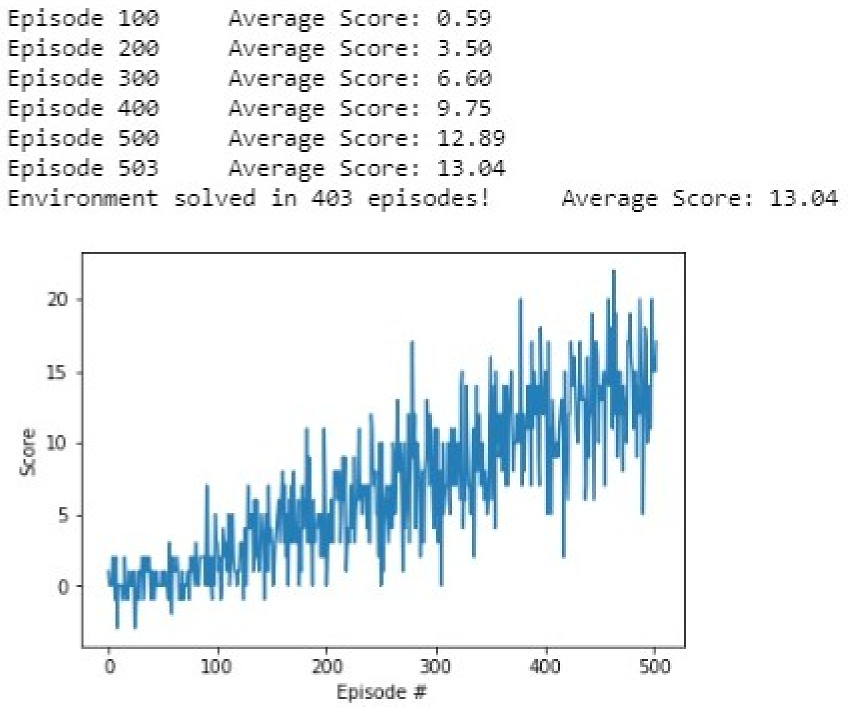
497 episodes

```
Episode 100     Average Score: 0.59
Episode 200     Average Score: 3.50
Episode 300     Average Score: 6.60
Episode 400     Average Score: 9.75
Episode 500     Average Score: 12.89
Episode 503     Average Score: 13.04
Environment solved in 403 episodes!     Average Score: 13.04
```

fc1: 64, fc2: 32, fc3: 16

403 episodes

Below is from the algorithm that I submit for this project.
With 3 fully connected layers (fc1=64, fc2=32, fc3=16), the environment was solved in 403 episodes.



```
Episode 100     Average Score: 0.59
Episode 200     Average Score: 3.50
Episode 300     Average Score: 6.60
Episode 400     Average Score: 9.75
Episode 500     Average Score: 12.89
Episode 503     Average Score: 13.04
Environment solved in 403 episodes!     Average Score: 13.04
```

Ideas for Future Work

Upon the completion fo this course, I will do some extras for this project:

- Make a video output of this project
- Improve the DQN algorithm performance using combination of:
  - Double DQN to tackle the over-estimate problem in Q-learning
  - Prioritized Experience Replay
    To prioritize experience replay, so that important experience transitions can be replay more frequently, to learn more efficiently.
  - Dueling DQN
    Combine state values and advantage values to estimate action value Q(s,a) function may result into faster convergence than vanilla Q-learning.
- There are other extensions, such as multi-step bootstrap targets, Distributional DQN, and Noisy DQN

References

1. Deep Reinforcement Learning Nano Degree Udacity Course

2. Deep Reinforcement Learning: An overview
   https://arvix.org/pdf/1701.07274.pdf

3. https://github.com/mgiammatteo/udacity_drlnd_project1/blob/master/Navigation.ipynb

4. https://github.com/xsankar/DQN_Navigation/blob/master/Report.pdf