Banana Navigation

For solving this environment, I used basic DQN neural network for training agent and predicting action against each different state.

Learning from this project

Input of the DQN is game state vector from Unity Agent and output of the DQN is action space for different state

a. Key Point about DQN

1. Fully-connected layer - input: 37 (state size) output: 64
2. Hidden Fully-connected layer - input: 64 (state size) output: 64
3. Fully-connected layer - input: 64 output: (action size)
4. Maximum steps per episode: 1000
5. Starting epsilon: 1.0
6. Ending epsilon: 0.01
7. Epsilon decay rate: 0.999
8. Update neural network after each 4 step and batch size of 64
9. Discount factor: 0.99

Plot of Rewards

Plotting Average Score against Episode of the game.

Basic DQN

1. Training Phase

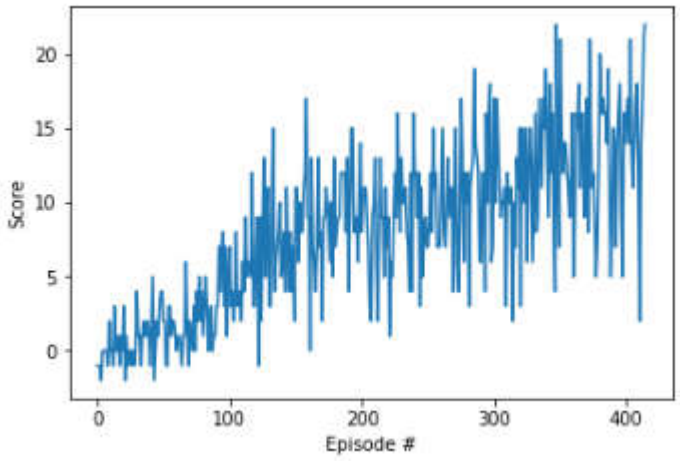
```
In [11]: scores = dqn(agent,n_episodes=4000,train=True,checkpoint_filename='dqn_checkpoint.pth')

Agent training is enabled.
Episode 100     Average Score: 1.43
Episode 200     Average Score: 7.67
Episode 300     Average Score: 9.49
Episode 400     Average Score: 12.28

Environment solved in 315 episodes!     Average Score: 13.00
neural network training weight is stored in file 'dqn_checkpoint.pth'
Score: 22.0
```

2. Production Phase

```
Loading Agent with weights from file 'dqn_checkpoint.pth'.
Agent is running for production.
Episode 100     Average Score: 15.14
Score: 22.0
```



Future Implementations

1. Dueling DQN
2. Double Deep Q Networks with Prioritized Experience Replay
3. Rainbow