P1 Navigation Project Report

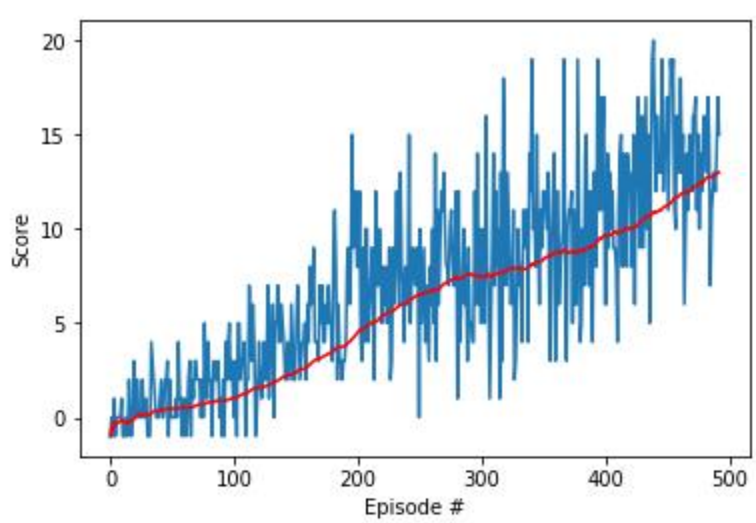
Implementation:

- The learning algorithm is DQN, a value based RL learning algorithm. The following are the code structure of the implementation.
    - File dan_agent.py includes definition of Agent and ReplayBuffer class.
    - File model.py includes definition of neural network structure of DQN
    - File Navigation.ipynb includes the main training function dqn which will run a training procedure to complete the training
    - File checkpoint.pth included a trained model.

Learning Algorithm and Plot of Rewards:

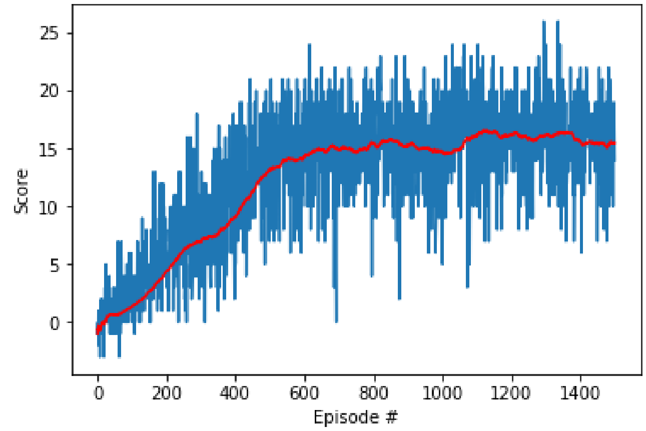
- The DQN agent model is a neural network which includes following layers
    - 3 Linear (Dense) layers
    - 1st and 2nd linear layers are followed by relu activation layer
    - 3rd linear layer is output layer which outputs the value for each stage-action pair
- Here are the hyperparameters used in 1st training experiment and experiment results.
    - eps start=1.0, eps end=0.01, eps decay=0.995
    - BATCH_SIZE = 64, GAMMA = 0.99, TAU = 1e-3, LR = 5e-4, UPDATE_EVERY = 4

```
Episode 100     Average Score: 1.01
Episode 200     Average Score: 4.46
Episode 300     Average Score: 7.44
Episode 400     Average Score: 9.65
Episode 492     Average Score: 13.00
Environment solved in 392 episodes!     Average Score: 13.00

<matplotlib.figure.Figure at 0x7f87f51f30b8>
```



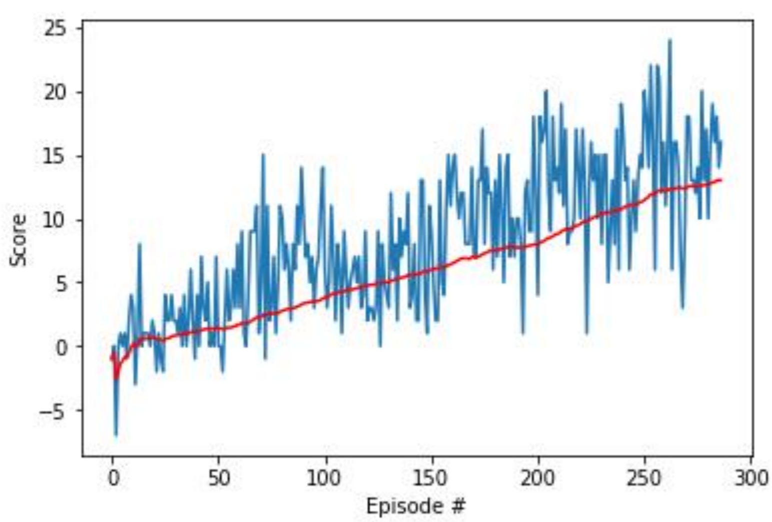
- Change the threshold from 13.0 to 20.0

```
Episode 100	Average Score: 1.33
Episode 200	Average Score: 4.35
Episode 300	Average Score: 6.90
Episode 400	Average Score: 9.12
Episode 500	Average Score: 13.10
Episode 600	Average Score: 14.36
Episode 700	Average Score: 14.86
Episode 800	Average Score: 15.45
Episode 900	Average Score: 15.34
Episode 1000	Average Score: 14.74
Episode 1100	Average Score: 16.03
Episode 1200	Average Score: 16.28
Episode 1300	Average Score: 16.17
Episode 1400	Average Score: 15.57
Episode 1500	Average Score: 15.49

<matplotlib.figure.Figure at 0x7fa3bc4672b0>
```



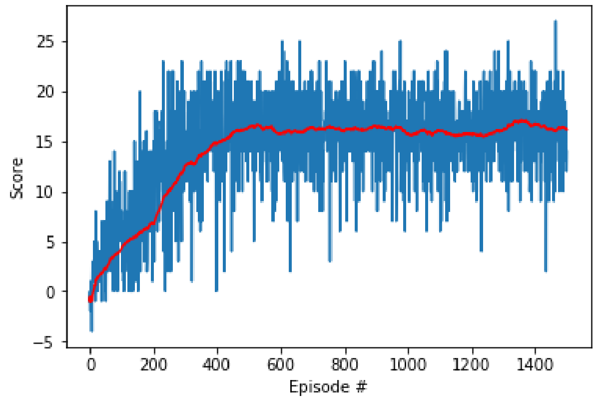
- Here are the hyperparameters used in 2nd training experiment and experiment results.
    - eps start=1.0, eps end=0.01, eps decay=0.5
    - BATCH_SIZE = 64, GAMMA = 0.99, TAU = 1e-3, LR = 5e-4, UPDATE_EVERY = 4
-

```
Episode 100	Average Score: 3.72
Episode 200	Average Score: 7.95
Episode 287	Average Score: 13.00
Environment solved in 187 episodes!	Average Score: 13.00

<matplotlib.figure.Figure at 0x7fb012d63080>
```



- Increase the stop threshold from 13.0 to 20.0. The training saturates at average reward 15.0

```
Episode 100	Average Score: 4.21
Episode 200	Average Score: 6.75
Episode 300	Average Score: 12.54
Episode 400	Average Score: 14.76
Episode 500	Average Score: 16.41
Episode 600	Average Score: 15.79
Episode 700	Average Score: 16.36
Episode 800	Average Score: 16.01
Episode 900	Average Score: 16.49
Episode 1000	Average Score: 15.80
Episode 1100	Average Score: 16.01
Episode 1200	Average Score: 15.68
Episode 1300	Average Score: 16.08
Episode 1400	Average Score: 16.75
Episode 1500	Average Score: 16.15

<matplotlib.figure.Figure at 0x7f8b95135e48>
```



- Conclusion:
    - It seems that reducing eps decay, which accelerates decaying rate of exploration, will help the agent converge faster.
    - I also tested different learning rate and the result indicates that by increasing learning rate from 0.001 to 0.0005, the final convergence of the reward will increase from 15.0 to 16.0

Learning Algorithm and Plot of Rewards:

- Change DQN to DDQN
- Apply Prioritized Experience Replay
- Try Dueling DQN
- Apply Actor-Critic learning algorithm
- Increase layers of original DQN neural networks