**Project Report**

This report highlights the Q-Network, the learning agent along with the hyperparameters and provides some references for future work.

**Q-Network**

The network is a feed-forward network with 3 fully connected layers as follows

1. Input 37 (see state size) => Output 64
2. Input 64 => Output 64
3. Input 64 => Output 4 (action size)

with ReLU activation function that maps state -> action values.

**Learning Agent**

The learning agent is created as a class that interacts and learns from the environment with qnetwork\_local and qnetwork\_target as initialized QNetwork variants plus Adam variant as optimizer.

**Hyperparameters**

* Replay buffer size => 1e-5
* Minibatch size => 64
* Discount factor Gamma => 0.99
* TAU for soft update of target parameters => 1e-3
* Learning rate => 5e-4
* How often to update the network => 4
* Number of episodes => 2,000
* Max number of iterations per episode => 1,000
* Epsilon starting value => 1.0
* Epsilon min value => 0.01
* Epsilon decay rate => 0.995

**Training**

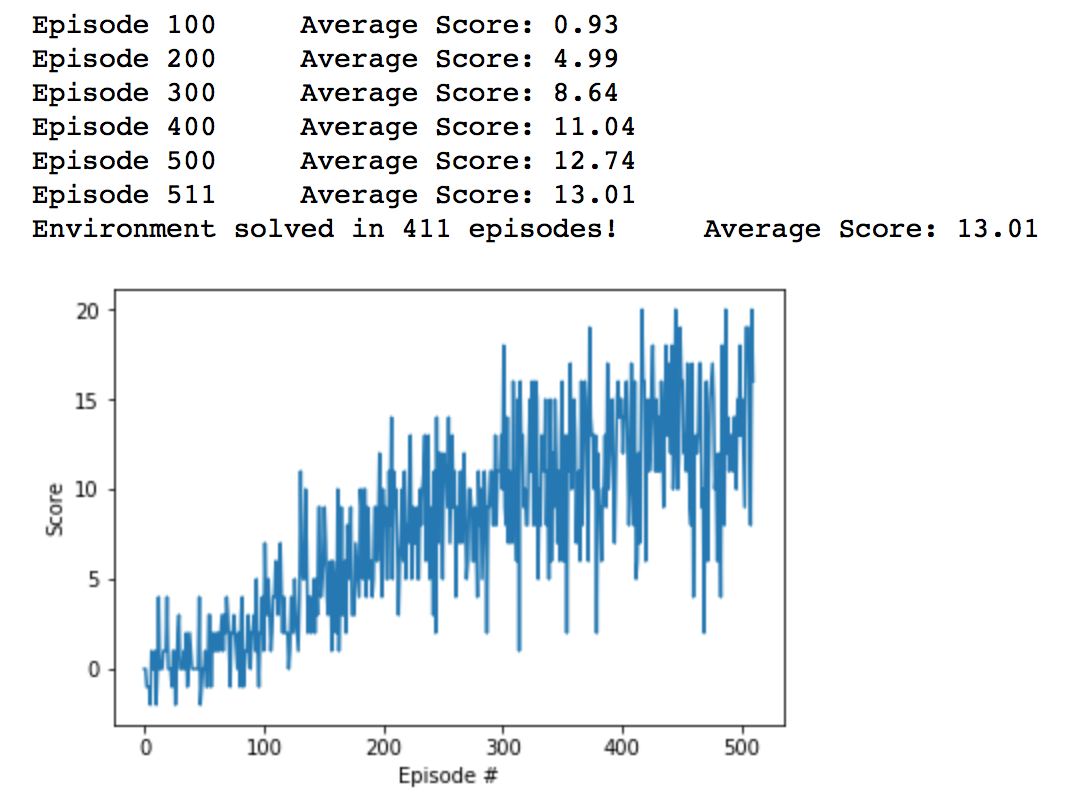
The training of the agent via dqn function (see Navigation\_Solution.ipynb) is running the following steps in every episode:

* Return actions for given state state as per current policy from qnetwork\_local using epsilon-greedy action selection
* Save experience in replay memory
* Learn every defined update time steps if enough samples are available in memory with random subset to update value parameters using given batch of experience tuples

1. Get max predicted Q values (for next states) from qnetwork\_target
2. Compute Q targets for current states
3. Get expected Q values from qnetwork\_local
4. Compute and minimize MSE loss
5. Soft update target network via

In total it is designed to run over 2,000 episodes (with 1,000 iterations per episode) - but it is considered as solved and hence ends if the agent gets an average score of +13 over 100 consecutive episodes.

**Plot of Rewards**



**Future Work**

Fine-tuning can be achieved by further hyperparameter optimization.

Additional enhancements can be achieved via implementing

* Double Q-Learning (see <https://arxiv.org/abs/1509.06461>)
* Dueling Q-Networks (see <https://arxiv.org/abs/1511.06581>)
* Prioritized Experience Replay (see <https://arxiv.org/abs/1511.05952>)
* Rainbow Approach (see <https://arxiv.org/pdf/1710.02298.pdf>)