Project 1: Navigation

To Solve this problem, the DQN algorithm was implemented using python3 and pytorch. The project consists of three main python:

1) Model.py
   Which holds the structure used for the target and local Q_Networks. The network consisted of 3 fully connected layers with the inner layers having 128 and 64 nurons respectively.

2) Replay_buffer.py
   This module represents the buffer where we store the experience tuples in order to select from it at random when the agent takes a learning step.

3) Navigation.py
   Has the implementation of the DQN including the act, step and learn function of the agent.

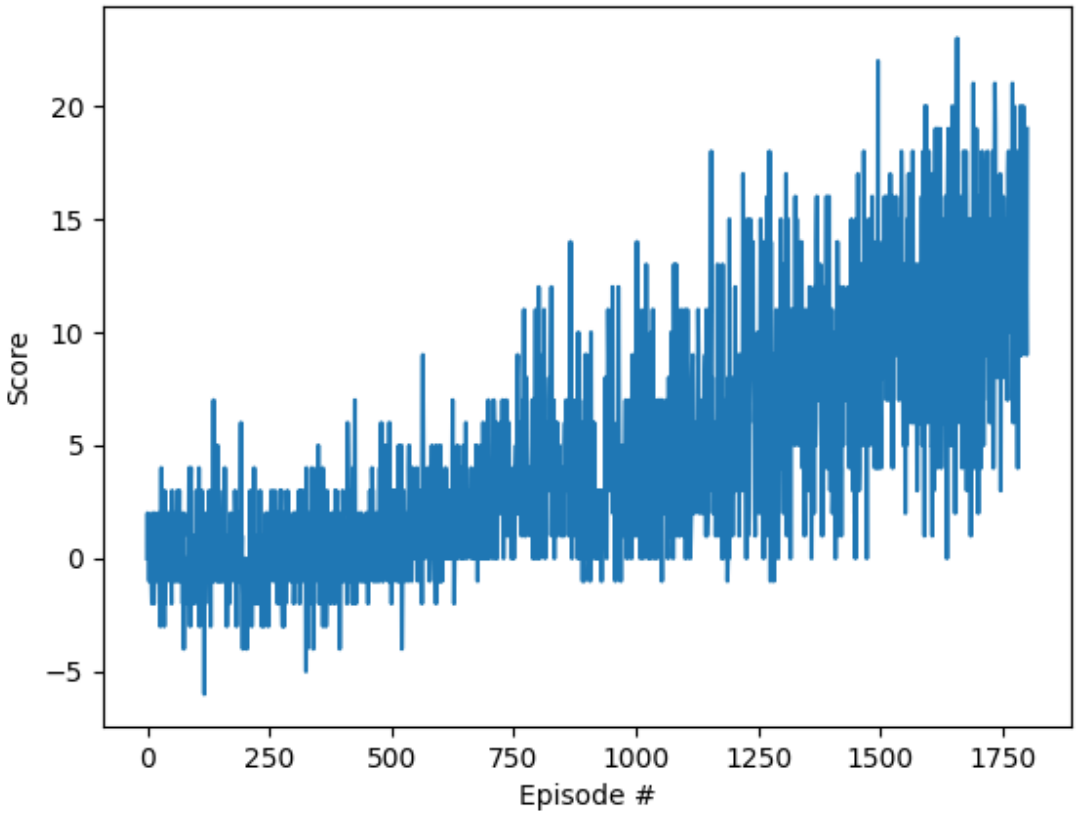
The chosen hyperparameters are:

```
BUFFER_SIZE = int(1e5)  # replay buffer size
BATCH_SIZE = 64         # minibatch size
GAMMA = 0.99            # discount factor
LR = 5e-4               # learning rate
UPDATE_EVERY = 4        # how often to update the local network
C = 10000               # how often to update the target Network
```

First I initialized the Q-Networks, the unity environment and the Replay buffer. Then I called the dqn() function which trains the agent for at least 2000 episodes until the agent successfully learns the environment by achieving 13 or more average points on the last 100 episodes.

In each episode the environment is reset and an initial state is observed and lasts at least 1000 time steps. In each time step an action is selected based on the current state following an epsilon-greedy policy, then the next state and reward are observed based on the chosen action. This new experience is then saved in the replay buffer by calling the step() function and if it is time to update the local network according to the `UPDATE_EVERY` variable, a learn step is taken where we sample 64 random experiences from the replay buffer and feed forward them to the local Q-network. The parameters of the local network are copied to the target Q-network every 10000 learning step.

Results:



As shown in the figure the agent was able to achieve an average score of 13 under 1800 episodes.

Future work:

I intend to modify the code by implementing the Prioritized Experience Replay and observe how it enhances the performance of the agent.