Learning Algorithm

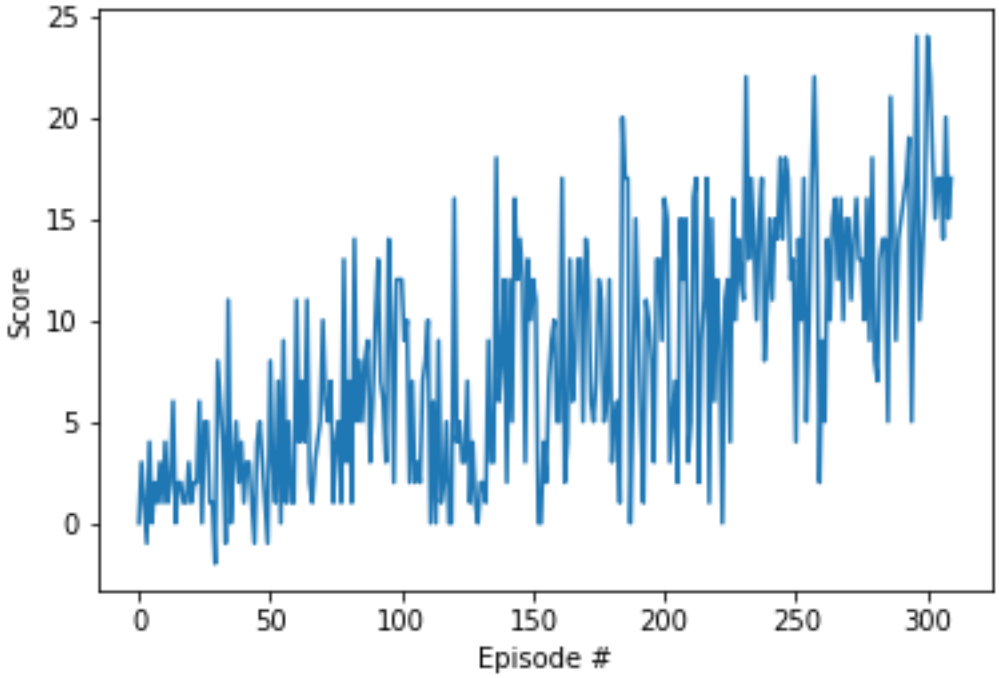
The algorithm used in this learning problem is deep Q-Learning.

A two-layer neural network (all fully-connected with rectified linear unit actions , each with 64 outputs) is applied as a non-linear function approximator (model architecture) for value-action pair. The input of the network is the observed states (37 in total) and this maps to the actions which the environment accepts (forward, backward, turn left and turn right). Experience replay is also used in order to mitigate oscillations often encountered in neural network while learning. A small batch of past experience is randomly sampled as the training goes on. Adam optimizer is used.

There are a number of hyperparameters which are tuned in the project. They are:

1. Replay buffer size
2. Batch size (minibatch size)
3. Discount factor
4. Learning rate
5. Update rate (how often to update the network), and
6. Epsilon values (initial, final and update)

Plot of Rewards



It took less than 400 episodes to score 13 on average. The final set of weights can be found in the repository and is named “checkpoint.pth”.

Ideas for Future Work

The demonstrated is quite similar to deep Q-learning algorithm from the classroom and it turns out it worked well for the navigation project as well. Other approaches such as double DQN and dueling DQN together with prioritized experience replay could be found working better as explained in Deep Reinforcement Learning lectures in terms of the number of episodes needed to solve.