PROJECT REPORT:

The project was a milestone which cemented my deep-RL understanding. I got a chance to tinker with various parameters as taught through the previous lessons.

STEP 1.
My first step was understanding the objective and nature of the problem. Used that to select the complexity of neural architecture and the size of the replay buffer.

STEP 2.
Maintained a record to find optimal hyper-parameters.

STEP 3.
Closely monitor the weighted average of the reward generation

STEP 4.
Freeze the computation graph and reiterate with a different architecture.

LEARNING ALGORITHM:

Value based Deep-Q learning was used for this project with the added advantage of a replay buffer with 150 prioritized experiences. However Double DQN was not used here.

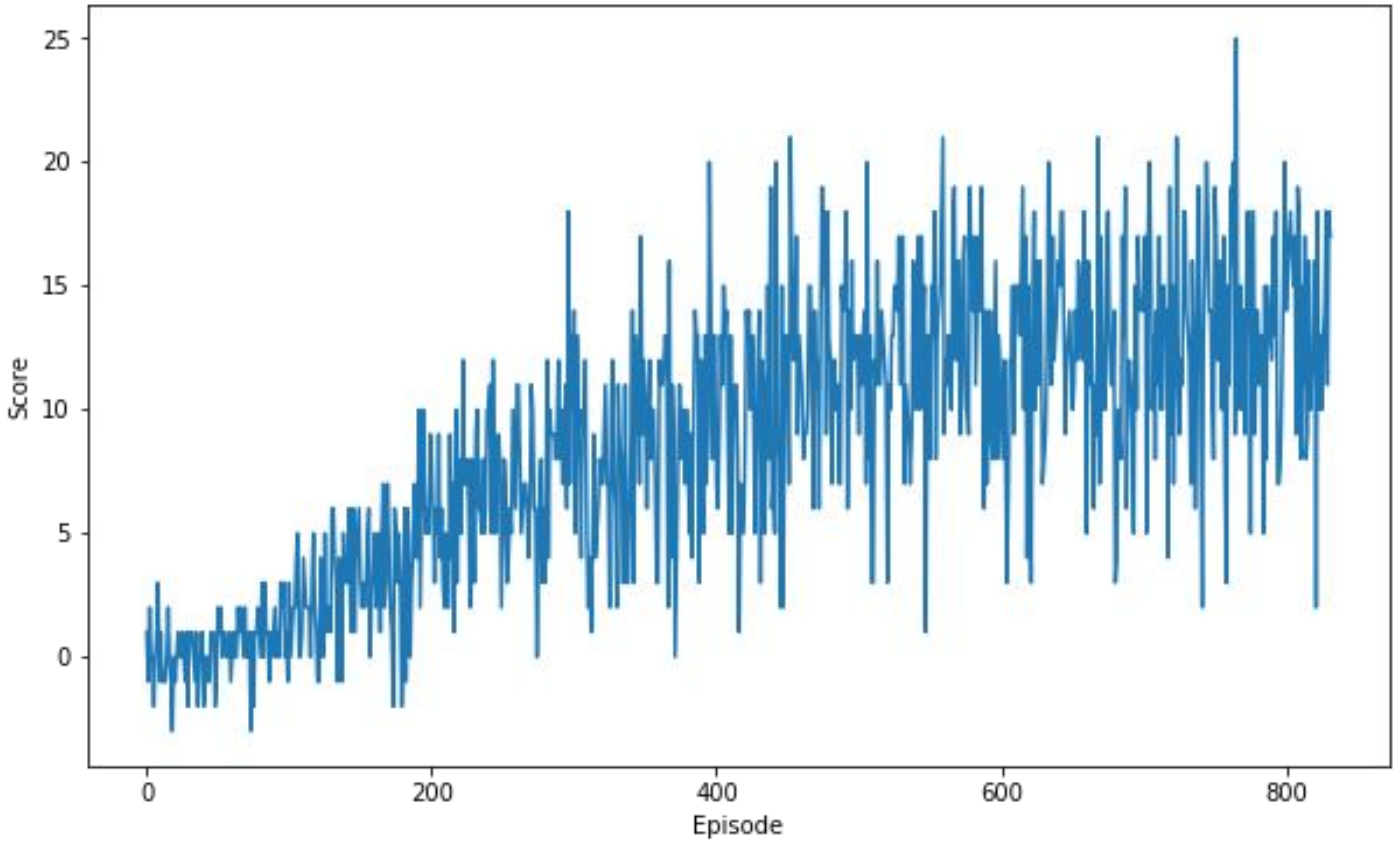
HYPERPARAMETERS:

The hyperparameters are:
n_episodes(number of episodes): 2000,
max_t(maximum time steps per episode): 1000,
eps_start(epsilon start): 1.0,
eps_end(epsilon end): 0.01,
eps_decay(decay constant): 0.995
scores_window_size(replay buffer size): 150
action_size(number of possible actions) : 4
state_size(state vector size): 37

MODEL ARCHITECTURE:

The neural network architecture is very simple. It consists of 2 hidden layers each with 64 nodes. The input layer is of size 37 nodes and output is of 4. ReLU is used as the activation function for the nodes. Model definition can be found in the model.py file of the project.

FINAL REWARD PLOT:



Conclusion:

I was able to solve the problem statement of attaining a moving average score of >13 in just 732 episodes. Was able to ensure convergence in minimal time.

Further Improvements:

1. A deeper model can be used to approximate value function.
2. The prioritization approach can be revisited to find out more informative experiences.
3. A policy-based approach can be placed in place of a value-based approach.
4. Bayesian optimization can be used to find out optimal hyperparameters.