# Project Report 1: Navigation

Rohan Sarkar

## Learning Algorithm

I have used the DQN and Double DQN Algorithm for training the agent. The 37-dimensional continuous state space observations are fed into the neural network shown below which learns an approximation of the Q value function for all the 4 actions in the discrete action space.

There are two separate Q-networks that are instantiated, which are:

1. Local Q-Network ($Q$)
2. Target Q-Network ($Q'$)

The most optimal action for any state $s_t$ is chosen using the Local Q-Network as :

$$max_a\, Q'(s_t, a)$$

The two algorithms differ in how the target Q values and subsequently the losses are computed.

For the DQN algorithm, the loss is computed as:

$$L = MSE[\, Q(s_t, a_t),\ r_t + \gamma\, max_a, Q'(s_{t+1}, a')])$$

For the Double DQN algorithm, the loss is computed as:

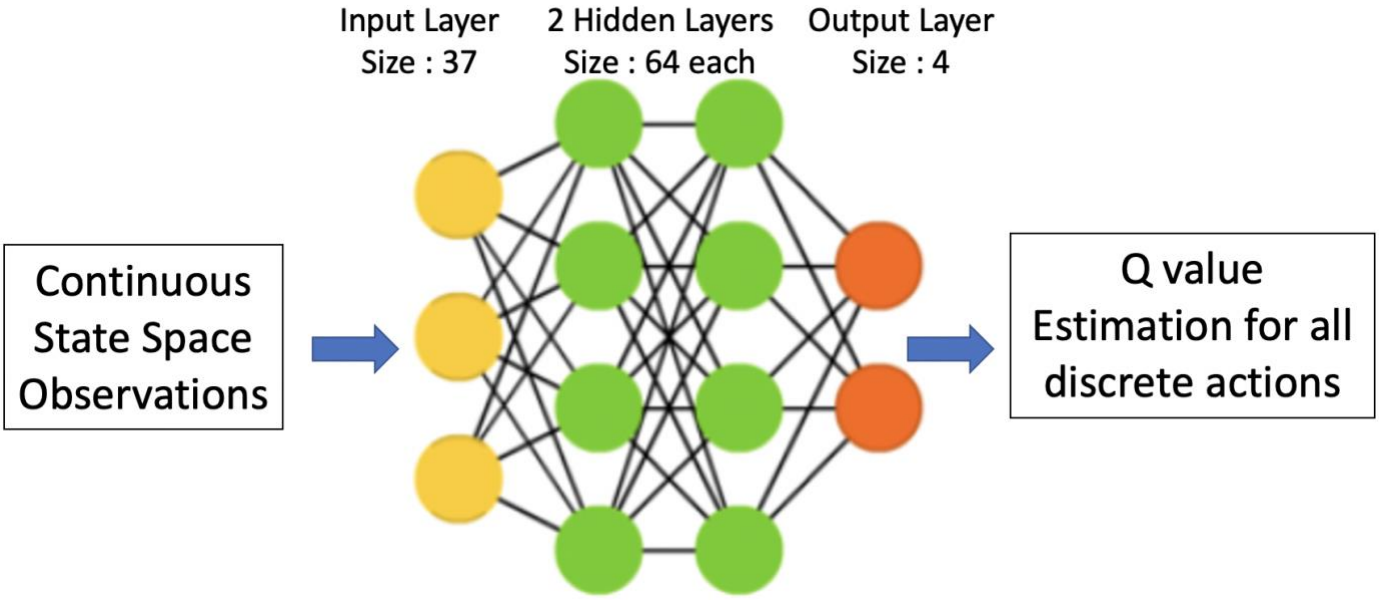
$$L = MSE[\, Q(s_t, a_t),\ \ r_t + \gamma\, Q'(s_{t+1}, argmax_a, Q(s_{t+1}, a')]$$

The Target network parameters are updated using a soft update strategy.

I use the epsilon-greedy approach. As the epsilon value decreases, the agent transitions from exploration to exploitation.

The Neural Network architecture and the hyperparameters are provided below.

## Neural Network Architecture:




## Hyper-parameters:

| Hyper-parameters | Description | Values |
|---|---|---|
| BUFFER_SIZE | Replay Buffer Size | 100,000 |
| BATCH_SIZE | Mini-Batch Size | 64 |
| GAMMA | Discount Factor | 0.99 |
| TAU | Soft update for target parameters | 0.001 |
| LR | Learning Rate | 0.0005 |
| UPDATE_EVERY | Interval to update the network | 4 |
| N_EPISODES | Maximum number of Episodes | 2000 |
| EPSILON_START | Starting value of epsilon for epsilon-greedy policy | 1 |
| EPSILON_END | Ending value of epsilon for epsilon-greedy policy | 0.01 |
| EPSILON_DECAY | Decay rate of epsilon | 0.995 |

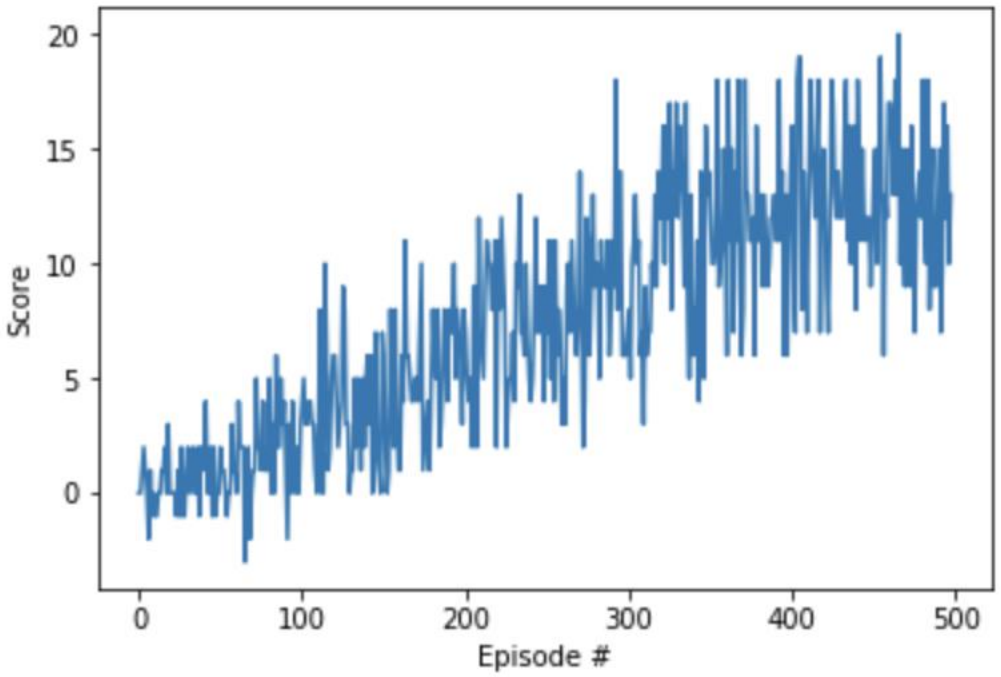
| | | |
|---|---|---|
| TARGET_SCORE | Average Target score to be reached for 100 consecutive episodes | 13.0 |

# Plot of Rewards

## DQN Results:

```
seed = 0
agent = Agent(state_size, action_size, seed, "DQN")
scores = train()
plot_scores(scores)
```

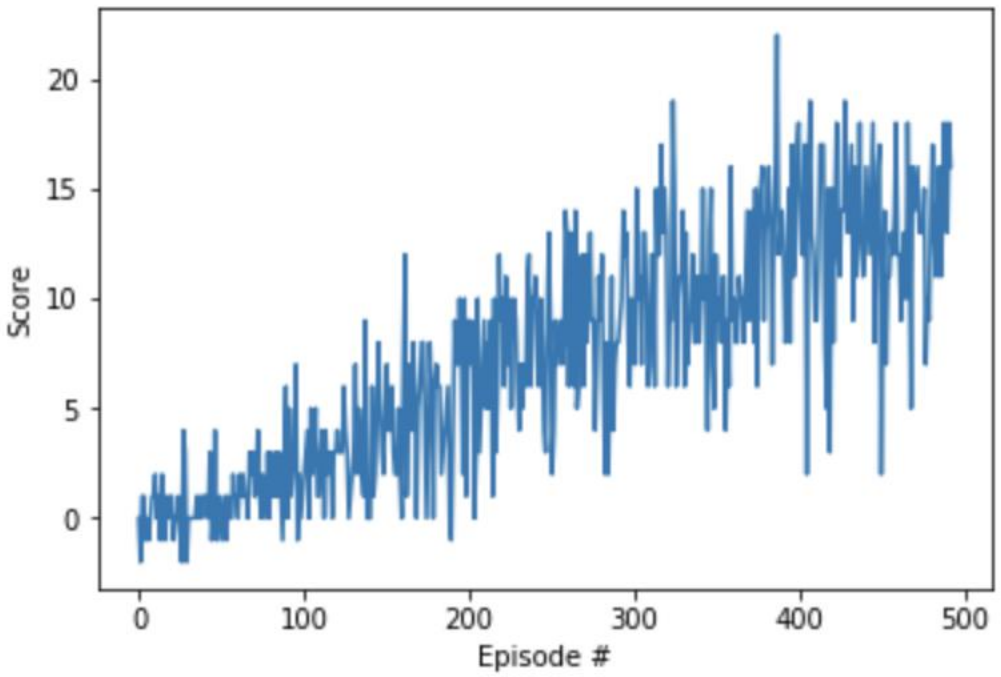
```
Episode 100     Average Score: 1.00
Episode 200     Average Score: 4.31
Episode 300     Average Score: 7.89
Episode 400     Average Score: 11.07
Episode 498     Average Score: 13.01
Environment solved in 398 episodes!     Average Score: 13.01
```




## Double DQN Results

```
seed = 0
agent = Agent(state_size, action_size, seed, "DDQN")
scores = train()
plot_scores(scores)
```

```
Episode 100     Average Score: 0.93
Episode 200     Average Score: 3.96
Episode 300     Average Score: 7.99
Episode 400     Average Score: 11.11
Episode 492     Average Score: 13.00
Environment solved in 392 episodes!     Average Score: 13.00
```




# Future Ideas

Future extensions would involve training the agent using Dueling DQN and replace the Replay memory with Prioritized Experience Replay.