# Report

For the navigation project, we used a DQN model to determine the best action at each step.

## Model architecture:

This model had a basic architecture of 2 fully connected neural networks with 64 nodes in each layer. The activation function used after each hidden layer is a RELU. The used architecture is the vanilla DQN model.

## Hyperparameters:

We initialized our variables to:

```
BUFFER_SIZE = 10000      # replay buffer size
BATCH_SIZE = 64          # minibatch size
GAMMA = 0.99             # discount factor
TAU = 1e-3               # for soft update of target parameters
LR = 5e-4                # learning rate
UPDATE_EVERY = 4
eps_start = 1.0          # starting value of epsilon, for epsilon-
greedy action selection
eps_end=0.01             # minimum value of epsilon
eps_decay=0.995          # multiplicative factor (per episode) for decreasing e
psilon
```

The maximum average score to reach before exiting the learning loop and save the checkpoint was initialized to 13.

Training the model with those variables had this output:

```
Episode 100      Average Score: 1.10
Episode 200      Average Score: 4.67
Episode 300      Average Score: 8.06
Episode 400      Average Score: 10.34
Episode 500      Average Score: 11.75
Episode 552      Average Score: 13.01
Environment solved in 452 episodes!      Average Score: 13.01
```
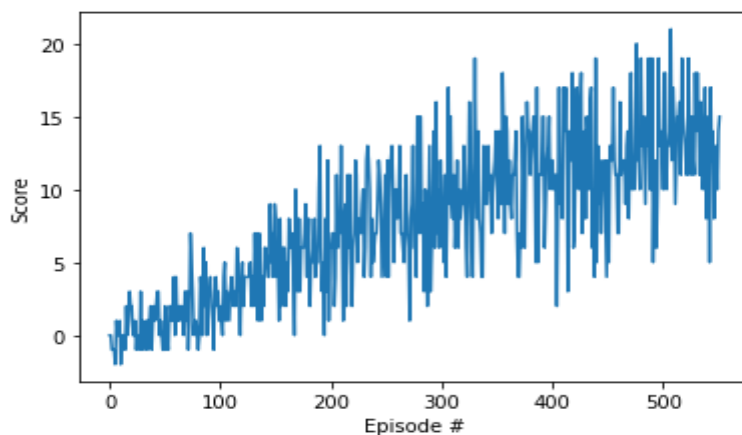
**Evaluation**

To evaluate our model, we made it run with eps = 0 to make sure it only applies its knowledge correctly. Running the model on 100 steps gave this result

```
Episode 100     Average Score: 15.43
```
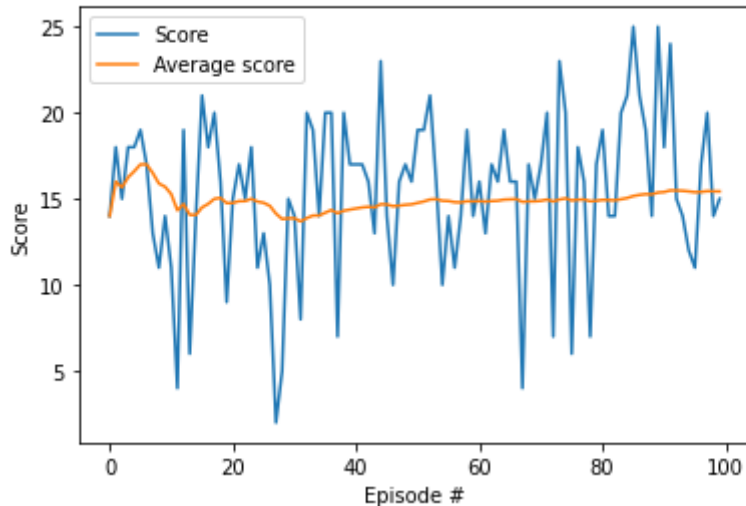


**Next steps**

The agent with a vanilla DQN was able to reach the 13 rewards but we got to aim higher. Trying a more complex model architecture could improve significantly the performance.

Here are the next step for this project:

- Add multiple models in different python files:

    - Rainbow: Combining Improvements in Deep Reinforcement Learning
    - Double DQN
    - Dueling DQN
    - Prioritized experience replay

- Make running a model configurable (choose the model to run from the config file)
- Make a scoring function based on the number of steps to reach a certain score and the best score a model can achieve to choose the best model.