# Navigation Udacity Deep RL Project Report

## Overview:

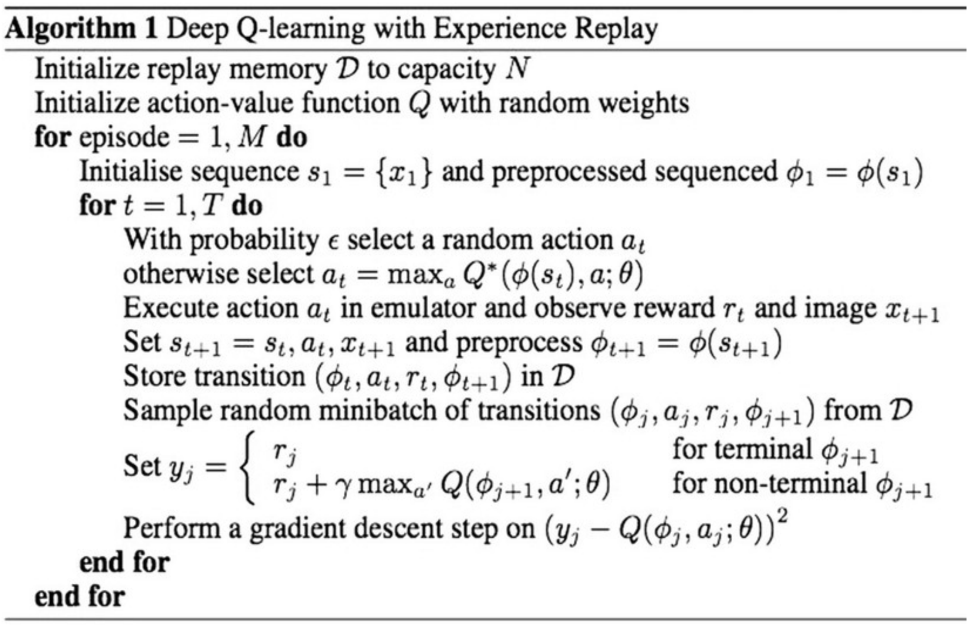
Through this report we will introduce the work done in the first project of the Udacity Deep RL course. We will present the project structure, after that the learning algorithm and finishing with showing the results.

## Project Structure:

- config.yaml: a configuration file where you could change the parameters.
- environment.yaml: contains the requirements used when creating the conda environment.
- main.py: the main file which is used to run the project.
- trainer.py: the trainer file which is used to train the agent.
- evaluator.py: the evaluator file which is used to evaluate the agent.
- model.py: the deep network model created with pytorch.
- agent.py: contains the class Agent.
- replay_buffer.py: contains the ReplayBuffer class used in the DQN algorithm.
- utils.py: contains the utils used in the project.
- model.pth: the saved weights of the trained model.

## Learning Algorithm:

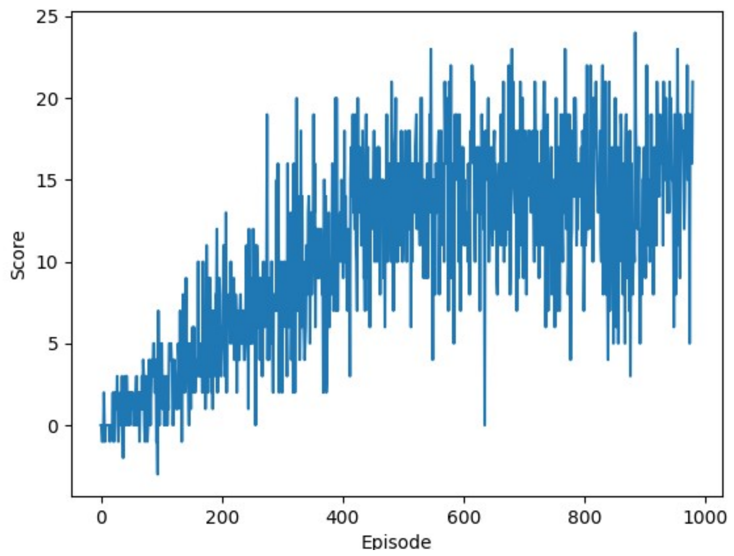
In order to train our agent we used the VanillaDQN algorithm illustrated in this pseudo-code:



```
**Algorithm 1** Deep Q-learning with Experience Replay
  Initialize replay memory $\mathcal{D}$ to capacity $N$
  Initialize action-value function $Q$ with random weights
  **for** episode $= 1, M$ **do**
      Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
      **for** $t = 1, T$ **do**
          With probability $\epsilon$ select a random action $a_t$
          otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
          Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$
          Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
          Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
          Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
          Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
          Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$
      **end for**
  **end for**
```

## Results:

This environment was solved after 900 episodes and you could see the plot in the figure below:


Score
Episode