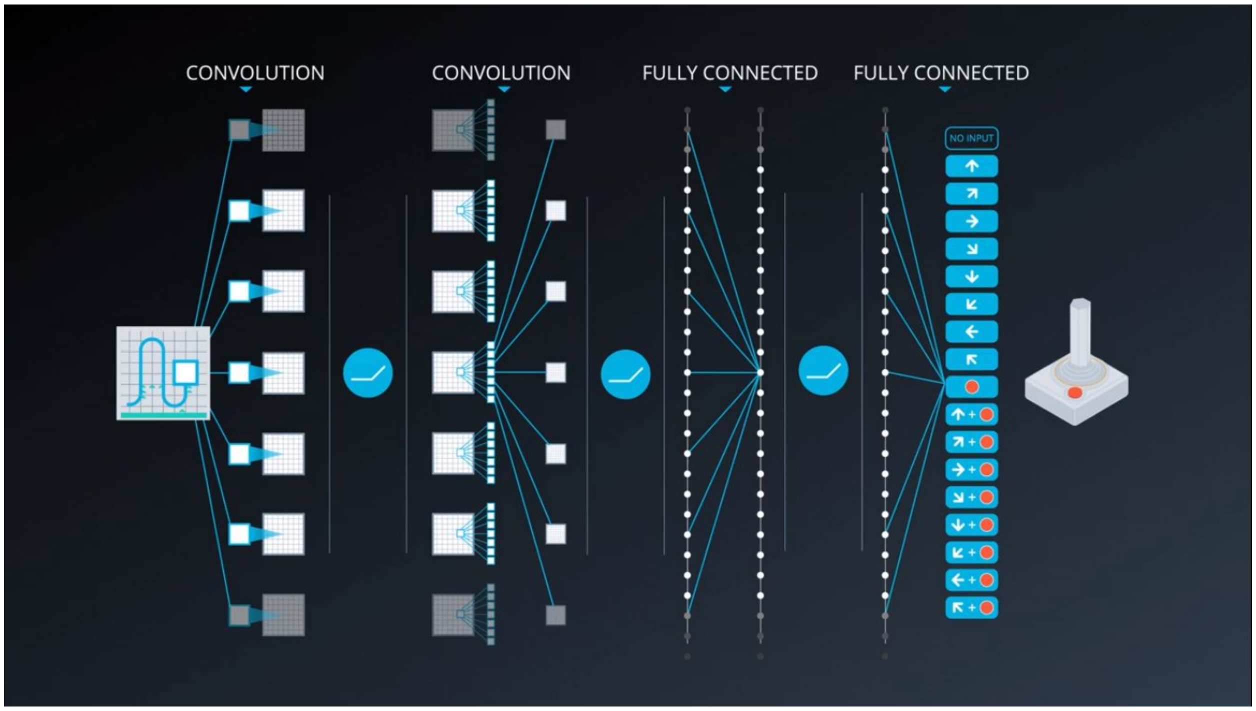
UDACITY

Project Navigation Report

By Taimur Zahid

Model Architecture and Algorithm: For this project, Deep Q-Network was used. The following image is a screenshot taken from one of the lessons of the Deep Reinforcement Learning Nandogree. The Model takes in an image as input and passes through a series of Convolution and ReLU Layers. It is then fed into a Fully Connected Layer which then returns an Action Value for each possible game action in one single pass instead of repeating it over and over again to produce an Action Value for particular game action.



For this project, the Deep Q-Network had only three Fully Connected Layers as shown in the code below. Note that this code was taken from the model.py file available in the GitHub Repository for this Project and this Nanodegree.

```
self.fc1 = nn.Linear(state_size, fc1_units)
self.fc2 = nn.Linear(fc1_units, fc2_units)
self.fc3 = nn.Linear(fc2_units, action_size)
```

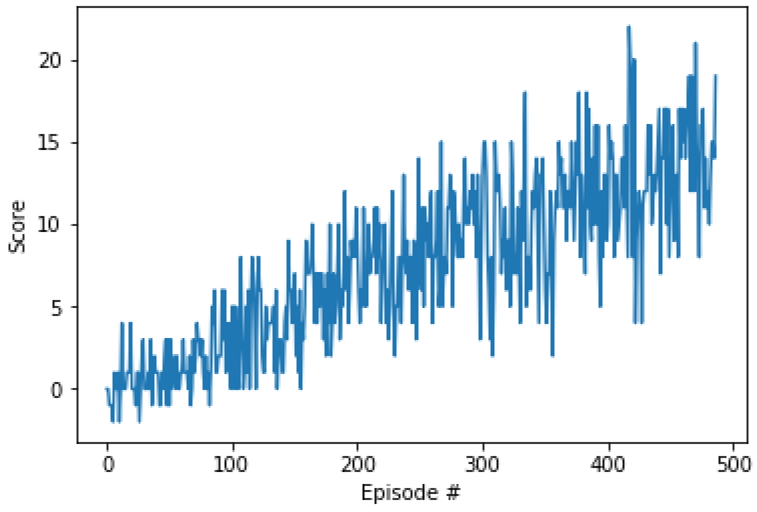
Hyperparameters: Most of the code used for this project is taken from the DQN Exercise Solutions Folder. The model was trained twice on two separate values of the Epsilon Decay. The Values for the Hyperparameters, taken from the dqn_agent.py and the Navigation.ipynb files, are as follows:

```
BUFFER_SIZE = int(1e5) # 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
```

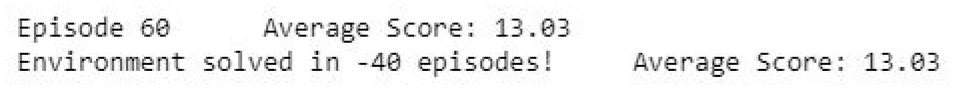
```
UPDATE_EVERY = 4 # how often to update the network
Random Seed = 0 # to specify the seed for the random value generator for the learning
phase.
n_episodes=2000 # maximum number of training episodes
max_t=1000 # maximum number of timesteps per episode
eps_start=1.0 # starting value of epsilon, for epsilon-greedy action selection
eps_end=0.01 # minimum value of epsilon
eps_decay=0.995 and 0.75 # multiplicative factor (per episode) for decreasing epsilon
checkpoint_number = 1 and 2 # specifying the name of the checkpoint file
```

Training Outputs and Plots: The model was trained twice using different values for the Epsilon Decay. The first approach had the Epsilon Decay set to 0.995. The Training output along with the graph are as follows:

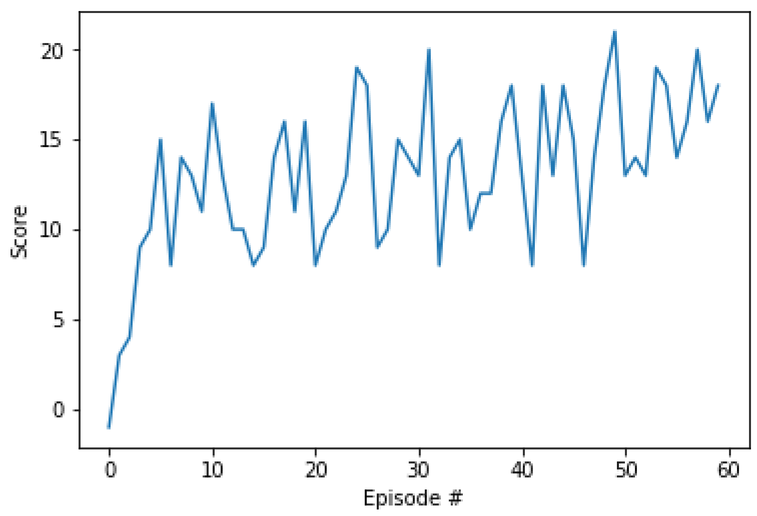
```
Episode 100	Average Score: 1.12
Episode 200	Average Score: 4.88
Episode 300	Average Score: 8.34
Episode 400	Average Score: 10.43
Episode 487	Average Score: 13.03
Environment solved in 387 episodes!	Average Score: 13.03
```



The second approach had the Epsilon Decay set to 0.75. The Training output and the graph are as follows:



```
Episode 60      Average Score: 13.03
Environment solved in -40 episodes!     Average Score: 13.03
```



Future Improvements: The following algorithms can be used to train a better model:

1. Double DQN
2. Dueling DQN
3. Distributional DQN
4. Noisy DQN