Learning Algorithm:

The Deep Q-Learning algorithm takes mainly two steps:

1) A sample step
2) A learn step

These steps are resume as follows:

To sample the environment a deep neural network is used as a nonlinear function approximator for calculate the value actions based on environment observations, the neural network maps the sensors data (environment states) (in this case a 37-dimensional vector of velocity values) to predicted actions values that maximize the future reward

For reduce the instabilities (oscillations) of neural network as a approximator the algorithm uses Experience replay, this allows take in account the history of experiences across episodes (a buffer (replay memory) that contains the last State S, the last Action A, the present Reward R and the present State S') in a stochastic way.

For a Learn step, a small batch of experiences is randomly sampled in order to train the agent, this breaks the correlations between consecutive experiences. The train process uses a gradient descent update step

Hyperparameters:

In the exercise there are a lot of parameters that would be tuned like a:

- Replay buffer size
- Minibatch size,
- the discount factor
- the learning rate
- update rate
- epsilon init value, final value, update value

other changes can be done in the system like a neural network architecture (number of layers, layers type, number of nodes in layers), the neural network optimizer (the Adam optimizer is used here), the random seed to the neural network of activation functions. For this particular problem, the best results are obtained by modifying the epsilon configuration for a init value of 1, final value of 0.0001 and decay value of 0.01, in fact I follow the suggestion in the lesson: Monte Carlo Methods->Exploration vs. Exploitation->Setting the Value of epsilon, in Practice.

Model Architecture:

The model architecture is defined through a few sets of variables as follows:

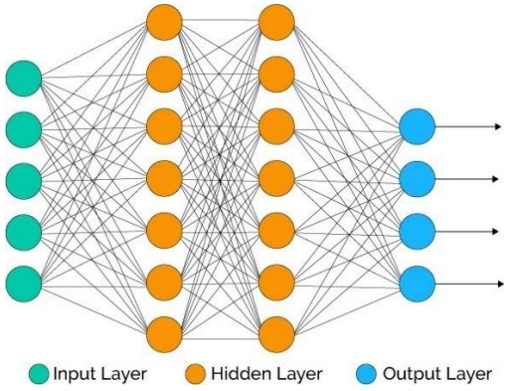
State_size represents the input dimensions of the network

Action_size represents the output dimensions of the network

Seed is used to initialize the weights of the network

fc1_units and fc2_units represent the number of nodes in the hidden layers of the network

For this Project the architecture is close represented as the next figure:



https://www.quora.com/What-do-you-mean-by-hidden-layer-in-neural-network

The input layer has 37 signals (signal of velocity), the hidden layers include 64 nodes and the output layer includes four signals (four actions)

Training:

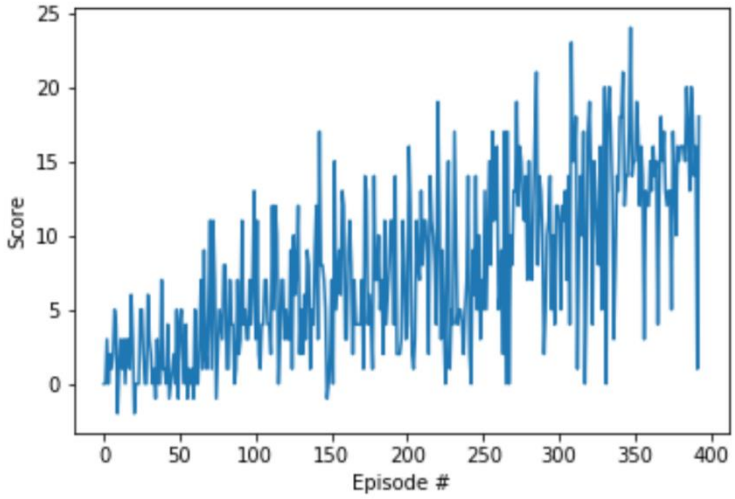
The training in **dqn** function runs with a max of 2000 episodes with 1000-time steps por episode. The epsilon-greedy action selection of takes the epsilon values defined in the hyperparameters section. The average of the 100 most recent scores determines the end of training process (if an average score equal to 13 in 100-time steps). Numerical lib Pytorch is used for his purpose. The training episode follows the steps below:

1) Get the agent action (with max likelihood) for present state and epsilon value
2) Get the environment action response that includes next state, reward and done flag
3) Update the agent with present state, present action, reward and next state
4) Update the state
5) Update the reward

At a final of each episode the epsilon value decrease by a factor and the neural network is trained. For this project the best result obtained is shown as follows:

```
Episode 100 Average Score: 2.76
Episode 200 Average Score: 6.15
Episode 300 Average Score: 9.05
Episode 393 Average Score: 13.00
Environment solved in 293 episodes! Average Score: 13.00
```

Plot of Rewards



The final weighs are included in either of two files named:

agentweights.pth or **checkpoint.pth**

Future Work:

For this project the lesson exercise code for lunar lander is adapted for unity learning agents, nevertheless as mentioned in the project Suggestions it's worth trying improvements like a double DQN, dueling DQN, or prioritized experience replay! Or all of them like a rainbow approach. The above can improve the score – episode performance by reduce the evident instability, also could reduce the total episodes as well as increase the total discounted reward.