Navigation Report

This report includes the description of the implementation of the Udacity Reinforcement Learning Navigation project.

Learning Algorithm, hyperparameters, model architectures

1. An agent is created with a deep learning (DL) model that get values of states and output values for each of the 4 actions. The process will train the deep learning model to generate the sets of weights for all layers.
2. Each step in an episode after an action is taken, from the current states, the environment generates next states, reward and whether the episode done or not.
3. The learning algorithm is to use the loss function to train the weights. The loss function is the mean squared error (MSE) of two results. One result is the max predicted Q-value generated by the DL model using next states as the inputs and then adjusted by discounting factor and added by rewards. Another result is the expected Q-value generated by the DL model using the current states as the inputs. The model is trained and updated by using back propagation on the loss function. This can be done at a given step interval.

$$L(\theta) = E\{(reward + \gamma * \max_{a'} Q(s', a'; \bar{\theta}) - Q(s, a; \theta))^2\}$$

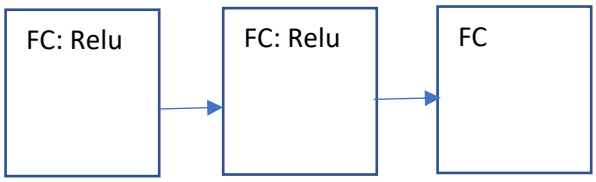
$$\Delta\theta = \alpha * \left(reward + \gamma * \max_{a'} Q(s', a'; \bar{\theta}) - Q(s, a; \theta)\right) * \nabla_\theta Q(s, a; \theta)$$

$$\bar{\theta} = \tau * \theta + (1 - \tau) * \bar{\theta}$$

4. The action is chosen to be the one corresponding to the maximum of the expected Q-value. In order to avoid the situation with local maximum, it needs to randomly choose action from time to time and eps is used to control how often to use random sample.
5. The hyperparameters are:

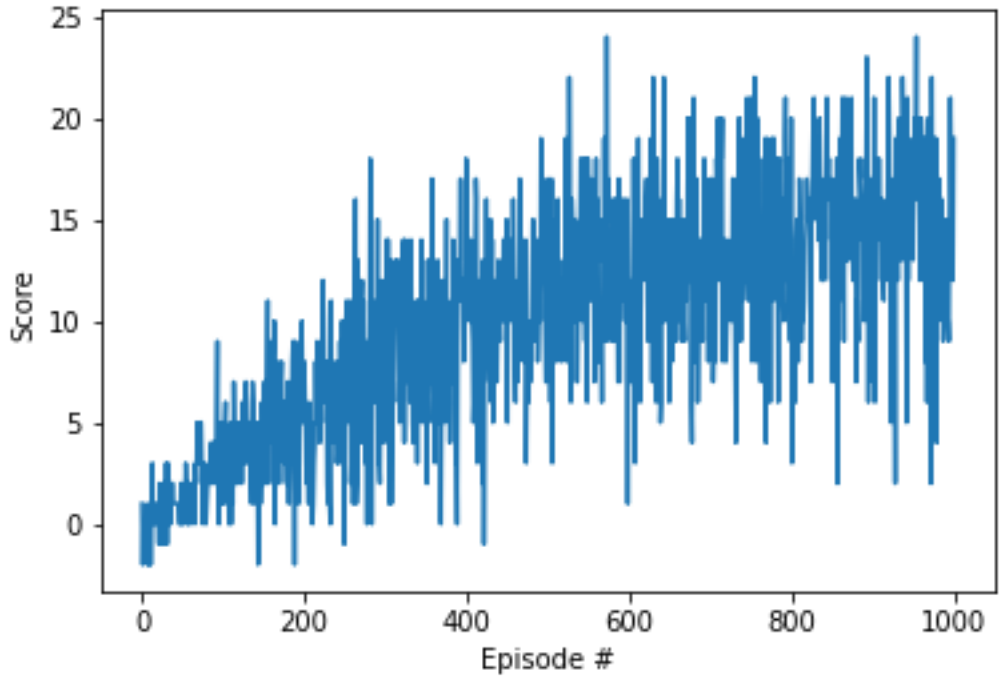
BUFFER_SIZE = int(1e5)	replay buffer size
BATCH_SIZE = 64	minibatch size
GAMMA = 0.99	discount factor
TAU = 1e-3	for soft update of target parameters
LR = 5e-4	learning rate
UPDATE_EVERY = 4	how often to update the network
eps_start = 1	
eps_end=0.01	
eps_decay=0.995	
eps = eps_start	

6. The DL model architectures for the neural networks are 3 fully connected (FC) layers are shown in the chart below. The weights are stored in the checkpoint.pth file.



Plot of Rewards

The plot of rewards per episode illustrates that the agent is able to receive an average reward (over 100 episodes) of at least +13. The number of episodes needed to solve the environment is 700.



Ideas for Future Work for improving the agent's performance

1. Use double deep Q network (DQN).
2. Use prioritized experience replay to train the model with more focus on those samples with larger deviation and avoid overfitting.
3. Use dueling DQN leading to better policy evaluation in the presence of many similar-valued actions.