Report: Navigation Project

Environment details:

A reward of +1 is provided for collecting a yellow banana, and a reward of -1 is provided for collecting a blue banana. Thus, the goal of agent is to collect as many yellow bananas as possible while avoiding blue bananas.

The state space has 37 dimensions and contains the agent's velocity, along with ray-based perception of objects around agent's forward direction. Given this information, the agent has to learn how to best select actions. Four discrete actions are available, corresponding to:

- **0** - move forward.
- **1** - move backward.
- **2** - turn left.
- **3** - turn right.

The task is episodic, and in order to solve the environment, agent must get an average score of +13 over 100 consecutive episodes.

Learning Algorithm:

Q-learning algorithm is used along with experienced replay. The optimization of network is done using Adam optimizer.

The network consists of three fully connected layers with Rectified Linear Unit (ReLU) activation functions between them.

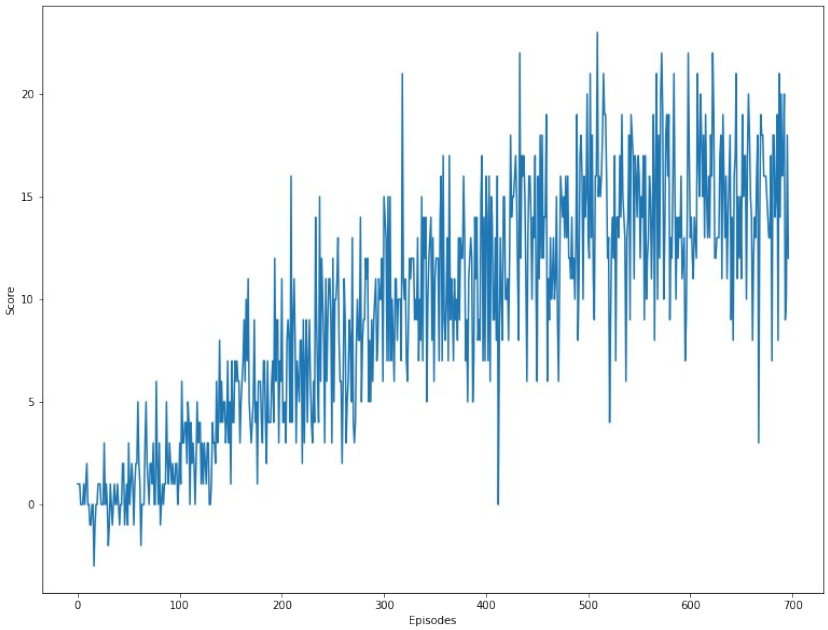
Layers:

fc1: fully connected layer (linear layer) mapping from state_size to fc1_units.
fc2: fully connected layer mapping from fc1_units to fc2_units.
fc3: fully connected layer mapping from fc2_units to action_size

Hyper-parameters:

BUFFER_SIZE = int(1e5)	replay buffer size
BATCH_SIZE = 64	mini batch 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_decay = .995	the reduction factor of the epsilon-greedy policy
eps_start = 1	initial value of epsilon
eps_end = .01	max value of epsilon

Plot of Rewards:



The environment is solved in 596 episodes with an average score of 15.02.

Ideas of Future Work:

1. I plan to integrate prioritized experience replay to this model. The bigger the error, the more we expect to learn from it. So, the error value is reflective of priority and is stored along with each corresponding tuple in the replay buffer. And while selecting samples from the replay buffer we use this value for guided selection.

2. This also needs further improvement as the samples with 0 error will never me selected. But this doesn't necessarily mean we have nothing more to learn from such a tuple. It might be the case that our estimate was closed due to the limited samples we visited till that point. To prevent tuples from being starved for selection, we can add a small constant e to every priority value.

3. Also completely relying on the priority weights can lead to only a small set of tuples being selected every time, resulting in a overfitting to that subset. To prevent this an element is introduced for uniform random sampling. This adds another hyperparameter (alpha) which is used to redefine the sample probability to make it more uniform.