Report

This project has three files – model.py, dqn_agent.py and navigation.ipynb

Code Implementation

1. model.py

The code used here is derived from the "Lunar Lander" project a part of Udacity's Deep Reinforcement Learning Nanodegree that used PyTorch to implement a Full connected Neural Network. The network is trained to predict the best action i.e. direction for agent to move towards the yellow banana.

2. dqn_agent.py

This code is also derived from the Lunar Lander project. To overcome the problem of DQN there are two popular methods - Replay buffer and Fixed Q
We have implemented Replay buffer where the Q value are retained in Q and is updated every 4 steps.
It has the following methods
constructor – to initialize the replay buffer and neural networks – local and target neural network
step – Store the state in replay buffer and update the target neural network after every 4 steps
act – retuans an action for a given state based on the current policy
learn – update the neural network weights
Replay Buffer
add – adds experience step to memory
sample – randomly sample a batch of experience for learning

3. navigation.ipynb

This code is similar to the one used in Lunar Lander but the difference is in the environment. Lunar Lander used OpenAI GYM while this banana navigator uses Unity3D. With a few changes to the navigation code it can be used for navigating the agent towards the yellow banana.
Jupyter Cell
[1] Import the necessary packages
[2] Examine the State and Action Spaces
[3] Take randam Actions in the enviroment
[4] Train the agent using DQN
[5] Plot the scores

Output

Model – checkpoint.pth
Plot - Episode Vs Score