Udacity Navigation Project

Introduction

This Project is given by Udacity Deep Reinforcement course. The aim is train an agent that can play a banana game. The aim is collect yellow bananas as much as possible with reward +1 point and if the agent collects blue bananas with reward -1 penalty point. The agent must provide an average score over of +15 over 100 consecutive episodes.

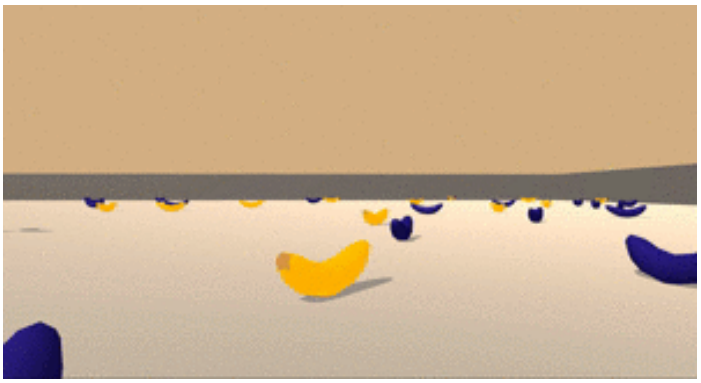


Figure1: An image of environment of bananas.

The environment has 4 actions, and 37 states size.

The agent trained for following scores:

Score 9 solved in 279 episodes in 785 seconds,

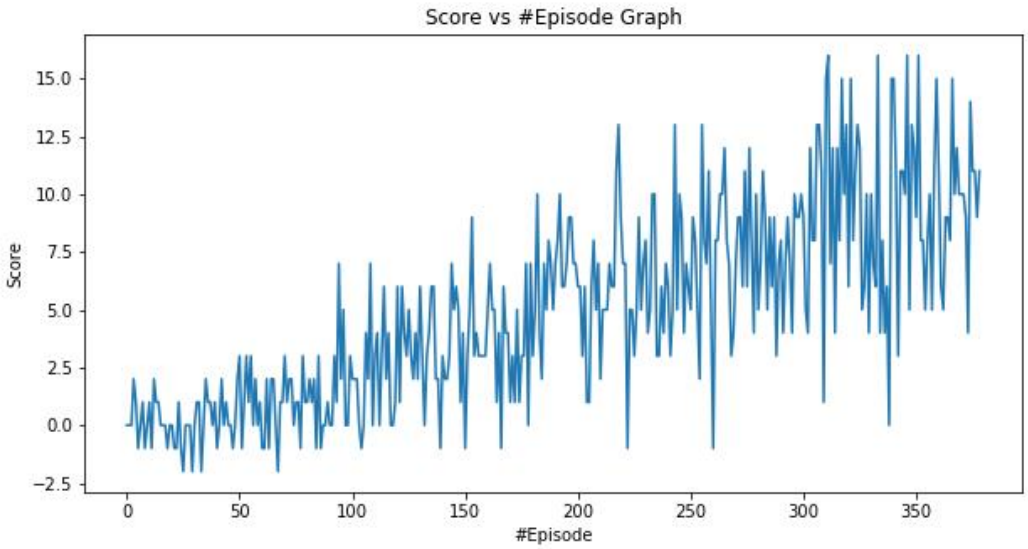


Figure 2: Graph of score 9

Score 11 solved in 332 episodes in 1208 seconds,

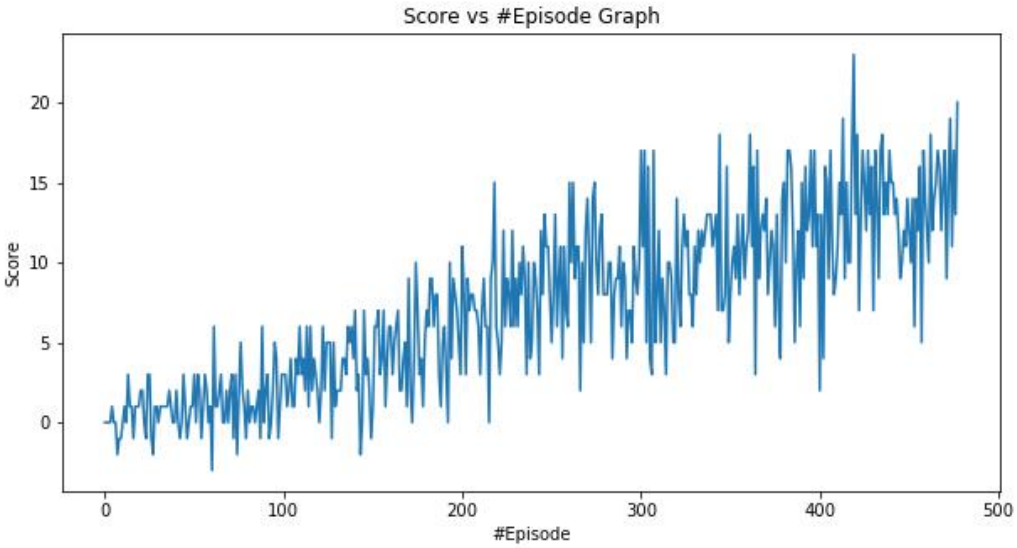


Figure 3: Graph of score 11

Score 13 solved in 378 episodes in 1329 seconds,

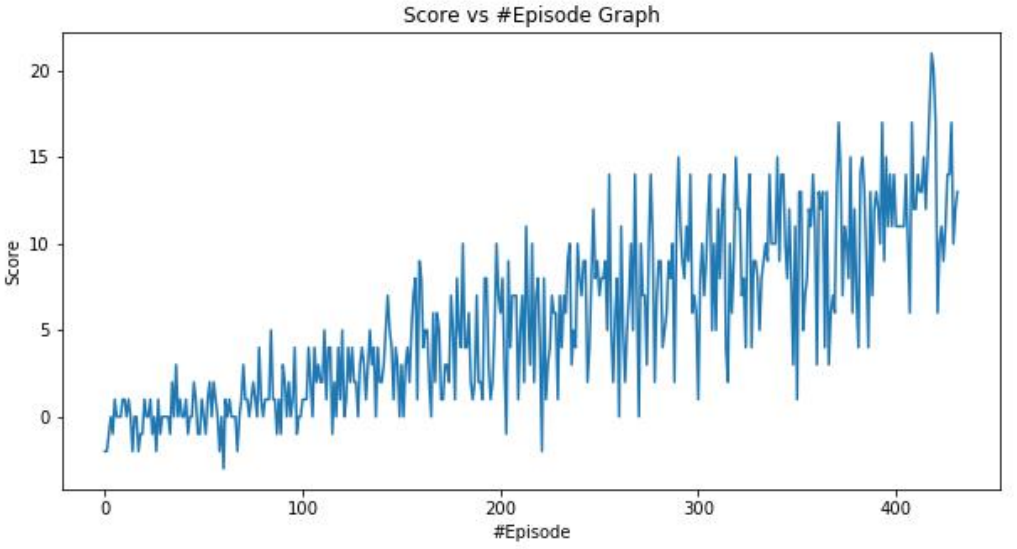


Figure 4: Graph of score 13

Score 15 solved in 664 episodes in 664 seconds,

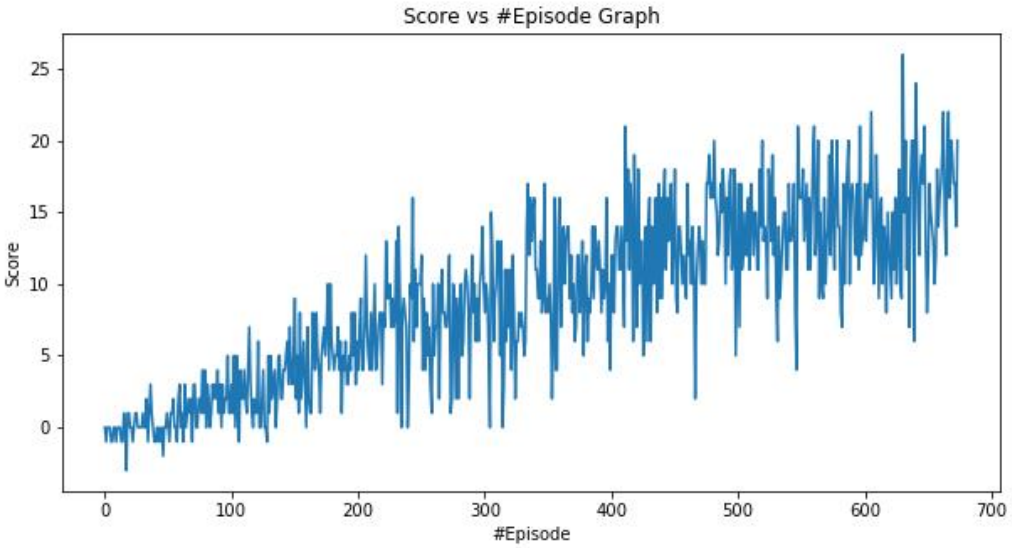


Figure 5: Graph of score 15

Score 17 cannot solve in 2000 episodes.

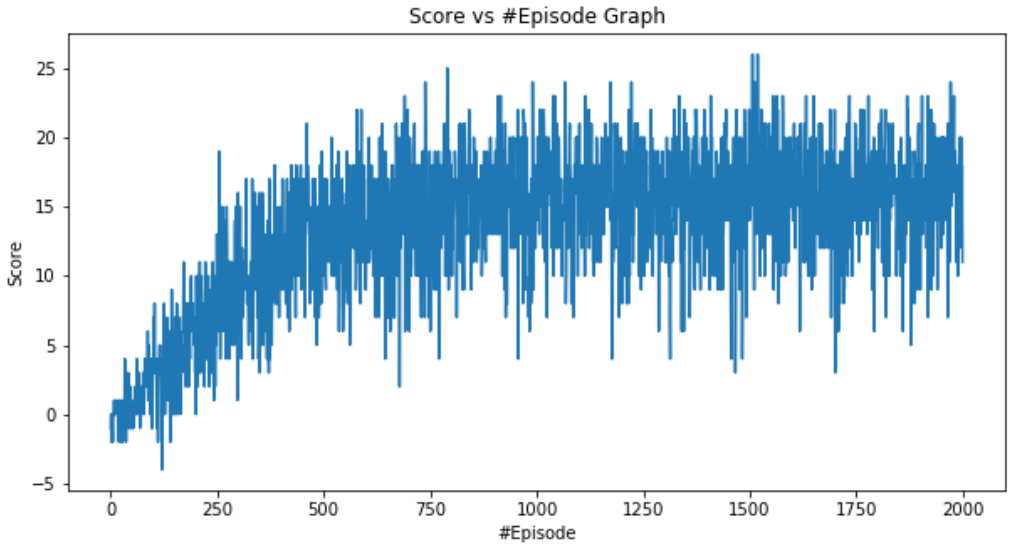
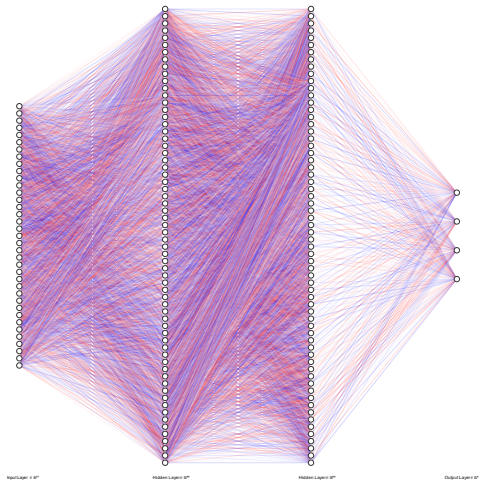


Figure 6: Graph of score 17

The best agent could be chosen from above by choosing best graph for required approach for our purpose, it is selected score +15.

Learning Algorithm & Model Architecture:

The DQN algorithm is used in the exercise. It has 3 layers. 37 inputs, 64 nodes and 64 node hidden layer and 4 action node.



The parameters are selected as in the exercise of dqn in lecture. However, it is required to find optimum point of these values since the environments are not the same and these values are set for hoover scenario. Both parameters and architecture of model could be changed. These will be done as during learning process as a future work.

These values are directly used from dqn_agent.py

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

Problems During Project:

The project was like the exercise in lecture but with different environments. So, it was easy to follow steps to finalize the project. The hard part was creating the environment to run the code written since there was always an error while importing some libraries. The most important one was torch library since it requires python 3.6. The other importing point was converting action to integer type otherwise the agent fails due to overflow.

Future Works:

- Parameter optimization:
  - The parameters are used directly from exercise but the environments are different so it is required to optimize these parameters.
- Different model architectures:
  - The model in the project is used directly from exercise so it could be tried different architectures for better result.
- Implementing double DQN and dueling DQN
  - Implementing these algorithms could be done as challenge and compare the results