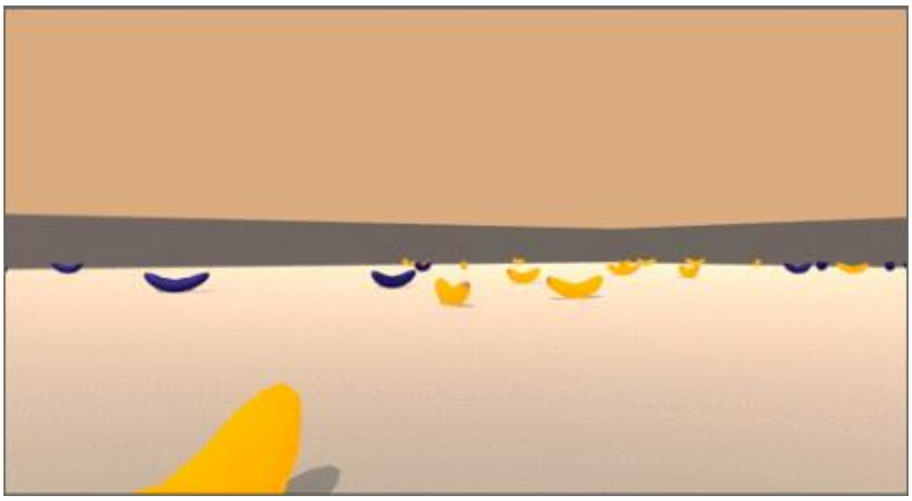
Udacity
Deep Reinforcement Learning Nanodegree
Project 1—Navigation
Liu Tianze

I. Introduction

In this project, I used DQN to train agent to navigate and collect yellow bananas in a large, square world.

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 my agent is to collect as many yellow bananas as possible while avoiding blue bananas.



II. DQN

For architecture, after many trial and error, I found that 3 FC layers could have better performance. In addition, I also found that using ELU as the activation function could have better results than ReLU because ELU was easier to converge and could avoid the dying-ReLU problem.

For neurons of FC layers, I have experimented many different combinations. For example, 512/256/128, 256/128/64, 128/64/32, and so on. I found that 128/64/32 could have the best results. Hence, too many neurons would not result in a better performance.

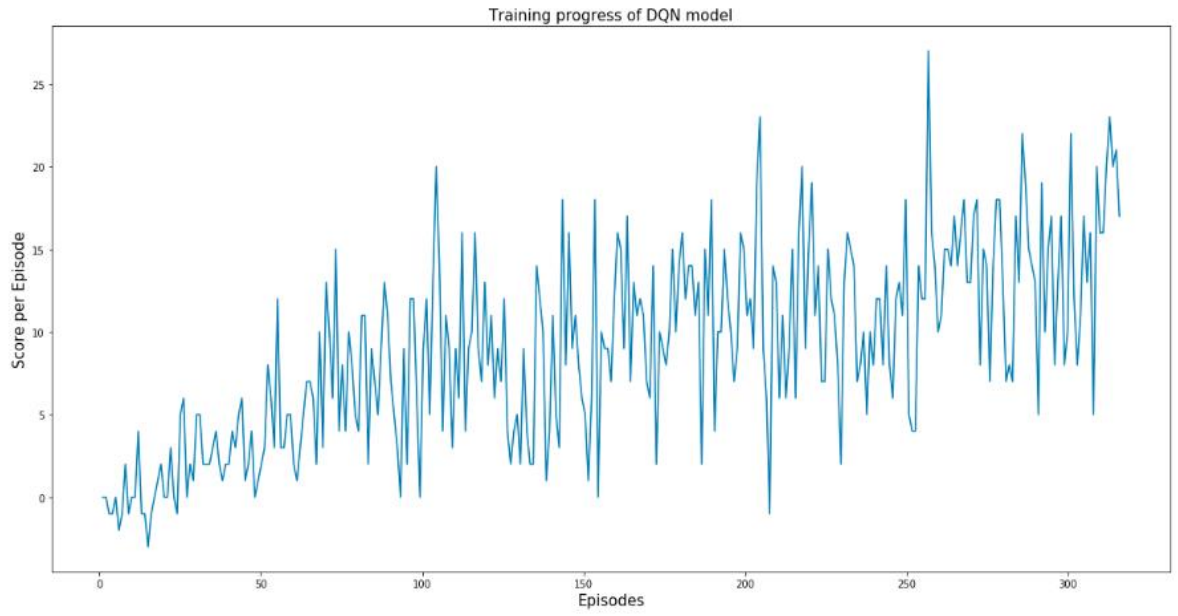
The agent used replay buffer as length 100,000 and it could have the ability hold important experiences during the training. After experimenting, I found batch size 64 was suitable for training. The soft update of target parameters (TAU) was 0.001. In addition, the DQN was updated every 4 steps and the learning rate was 0.0005. Furthermore, I wanted the agent to focus on the long term returns so the discount factor gamma was set to 0.99.

From the lecture videos, we knew that the choice of epsilon would have a big impact on the speed of convergence during training. After many trial and error, I selected epsilon starting at 1.0 and then decreased steadily with a decay rate at 0.93 during the training and stopped at 0.002 which was the minimum value of epsilon.

III. Results

From the results of my project, it achieved 13.00 score in average below 500 episodes. At first, the average score was around 0 or even negative. After about 100 episodes, the average score was closed to 4.00. After 300 episodes, the average score surpassed 10.00 in average. Finally, it achieved the average score of 13.06 at 315 episodes.

The training progress of DQN models was as follow:



IV. Further work

Although I have achieved the goal, there was still some room to improve. For example, I could use dueling DQN as my network and tried prioritized experience replay, maybe it would make me get better results. In addition, I could train the agent from raw pixels mentioned from optional challenge: Learning from Pixels.

Furthermore, after acquiring knowledge from the upcoming lectures of this nanodegree, I could use policy-based methods to solve this problem.