Solving Banana Collection in 3D Unity Environment by Deep Q-Learning

Xiongwen Qian

1. Problem

In this project, an agent is trained to navigate and collect bananas in a large, square world. The world is provided in a 3D Unity Environment. A reward of +1 is obtained after collecting a yellow banana, while a reward of -1 is obtained when collecting a blue banana.

The state space consists of 37 dimensions containing the agent's velocity and ray-based perception of objects around agent's forward direction. The agent's four discrete actions are:
0 – move forward
1 – move backward
2 – turn left
3 – turn right

The task is episodic. In each episode, the agent aims to collect as many as yellow bananas as possible to maximize the total reward obtained. An average score of +13 at least must be obtained over 100 consecutive episodes in order to solve the environment.

2. Method

The problem is solved by a Deep Q-Learning (DQN) method.

a) Q-Learning

A Q-Learning framework is adopted to solve the problem. Meanwhile, different from the standard Q-Learning algorithm, the action-value function $q_\pi(S, A)$ is approximated by an artificial neural network in the form $\hat{q}_\pi(S, A, w)$ in which the weight w is updated by the following formulas:

$$\Delta w = \alpha \left(R + \gamma \max_a \hat{q}_\pi(S', a, w) - \hat{q}_\pi(S, A, w) \right) \nabla_w \hat{q}_\pi(S, A, w)$$

$$w_{new} = w_{old} + \Delta w$$

where α is learning rate, R is reward, and γ is the discount rate.

b) Neural Network Structure

- The input is a state vector which contains 37 units.
- 2 hidden layers are used and each has 64 units. Activation functions are ReLu functions.
- The output layer has 4 units representing the action values for the 4 actions.

c) Experience Replay, Fixed Q-Targets

Experience Replay and Fixed Q-Targets techniques are used to enhance the learning algorithm. Experience tuples are saved and randomly retrieved when updating the weights of the neural network. The weights of the target Q-network are updated by the weights of the local Q-network according to the following formula:

$$w_{target} = \tau w_{local} + (1 - \tau) w_{target}$$

where τ is a hyperparameter to tune.

d) Hyperparameters

Learning rate $\alpha = 5e - 4$

Discount rate $\gamma = 0.99$

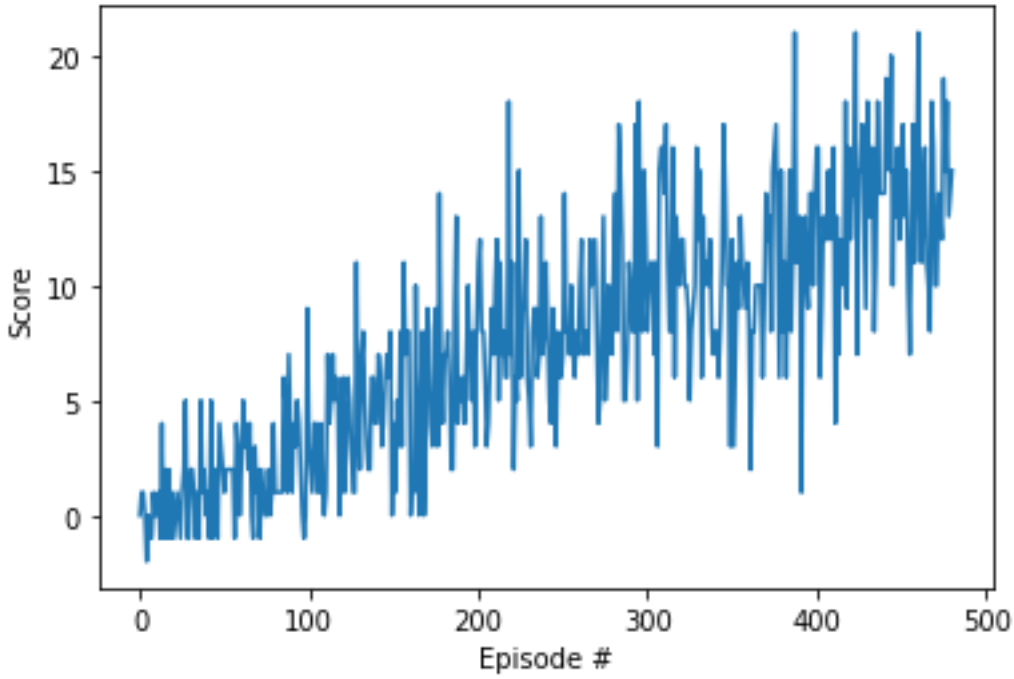
$\tau = 1e - 3$

The buffer size saving the experience tuple is $1e5$

Network weights are updated every 4 steps

3. Result

The environment can be solved in 500 episodes.



4. Future Work

Although the algorithm has achieved satisfactory result, several directions can be considered to further improve the work. First, currently the inputs are given states. A more natural way is to take the snapshot of the screen, (what the agent 'sees') as the input. It will require more advanced image preprocessing steps. Second, the hyperparameters may still have room to be fine-tuned to give a better performance.