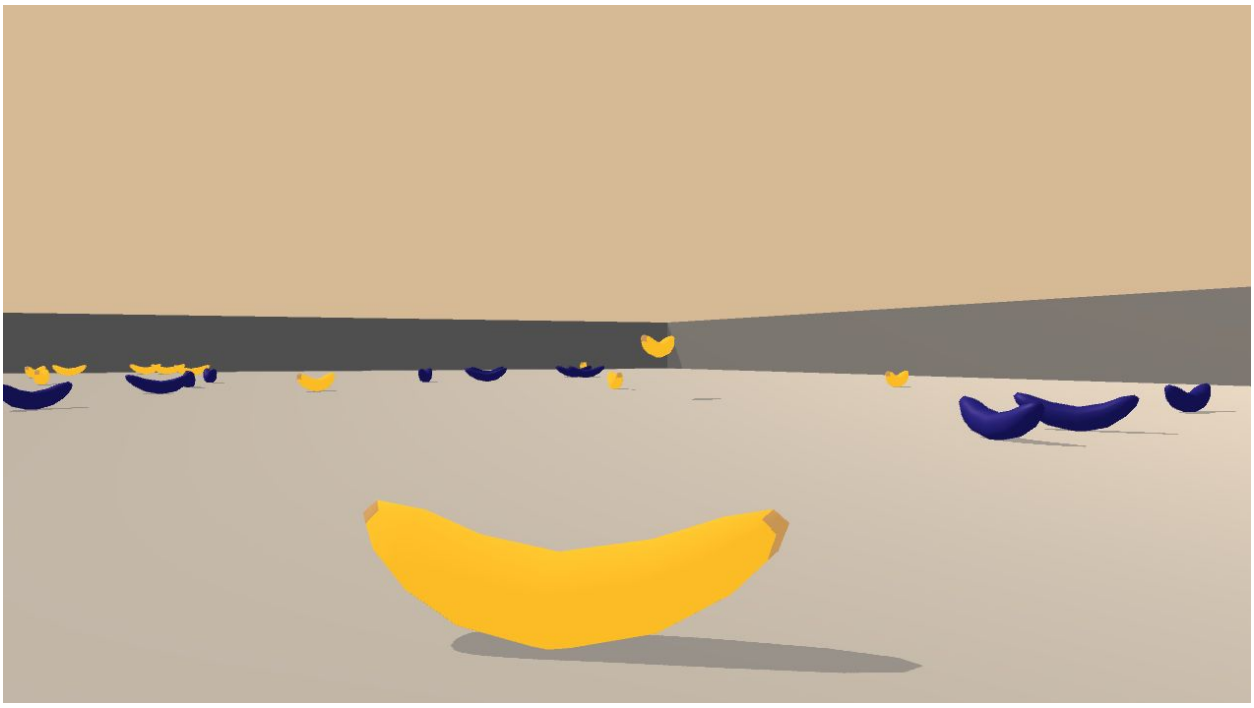


Navigation - Banana

By Rahil S. Vijay

Overview

The following report consists of a detailed analysis of my DQN implementation on Banana environment from Unity Environment

Environment

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 your 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 a ray-based perception of objects around the agent's forward direction. Given this information, the agent has to learn how to best select actions. Four discrete actions are available, corresponding to

1) 0-*move forward*

2) 1-*move backward*

3) 2-*turn left*

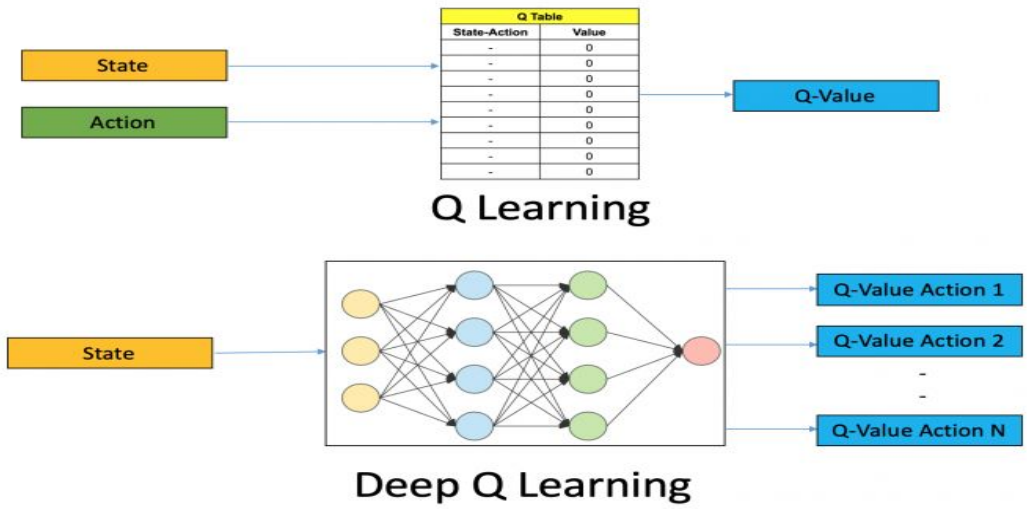
4) 3-*turn right*.

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

Algorithm

DQN

In deep Q-learning, we use a neural network to approximate the Q-value function. The state is given as the input and the Q-value of all possible actions is generated as the output. The comparison between Q-learning & deep Q-learning is wonderfully illustrated below:



The equation that we use to update the Q-learning is :

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right]$$

The equation that we used to update the neural network in Deep Q-Learning is

$$\Delta \mathbf{w} = \alpha\Big(\underbrace{R + \gamma \max_a \hat{q}(S', a, \mathbf{w})}_{\text{TD target}} - \underbrace{\hat{q}(S, A, \mathbf{w})}_{\text{current value}}\Big)\nabla_{\mathbf{w}}\hat{q}(S, A, \mathbf{w})$$

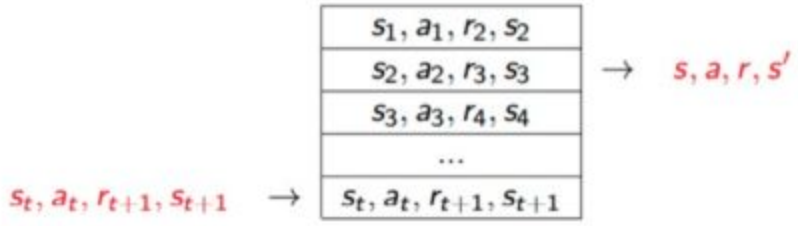
$$\text{TD error}$$

It has been observed that Deep Q-Learning is highly unstable. The original paper for *DQN*s addresses this problem. In order to avoid instability, they came up with two techniques:

Experience Replay

Experience tuples are of the form → (**State, Action, Reward, Next State, Done]**

When the agent interacts with the environment, it uses the sequence of experience tuples to train the agent. The problem with this is that these sequenced tuples can be highly correlated, as the action you take from a particular state effects the next state. Hence in order to remove this correlation, we use a replay buffer that stores all the experience tuples. Once enough experiences are gathered in the replay buffer, we then use it to sample a batch of experiences at every time step to train the agent.



Replay Buffer – fixed size

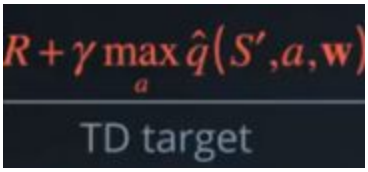
In my implementation, I have used a replay buffer of 100000.

Fixed Q- Targets

In DQNs in order to compute the TD error, we need to calculate the TD target and the currently predicted Q estimation.

As we don't have the real TD target, we end up estimating it. Resulting in the problem that we use the same network to estimate the TD target and the Q estimation. As a consequence, there is a big correlation between the changing parameters(weights, biases in the Q-network) and the TD target.

This means every time our Q-network shifts the TD target also shift as we use Q-network for estimating it.



To remove this correlation, we use another network(called the target network) to estimate the TD target. We update this target network after every t time step(in my project it was 4).

This is how are new loss function looks for Q-network. Here the w- is updated after every t time step, keeping the loss function stable.

In my implementation, I have also added two more techniques that help DQNs to stabilize.

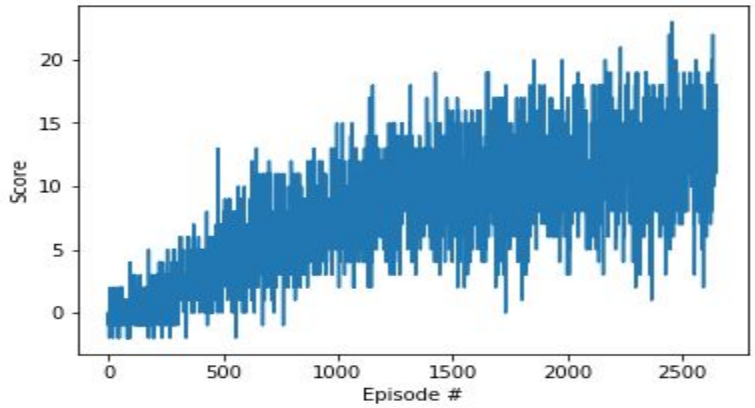
Double DQN - Link

Duelling Network - Link

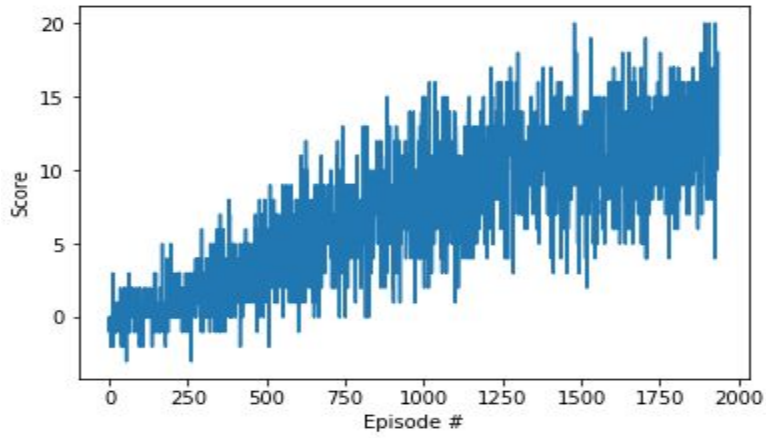
You can read more about them in their respective links.

Results

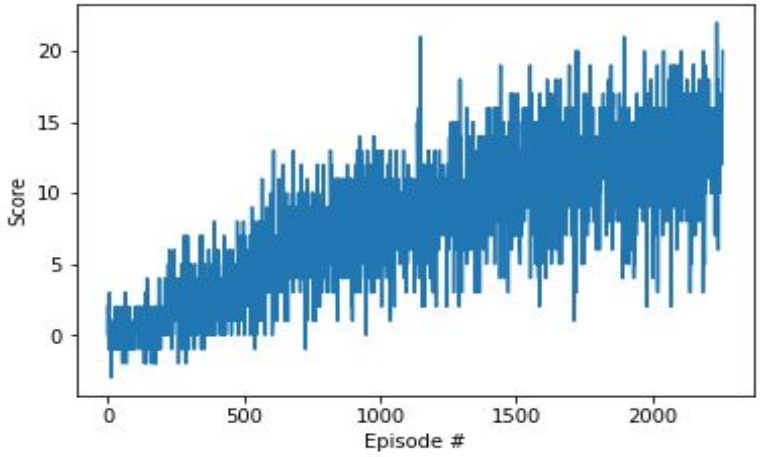
DQN



DDQN



DDQN + Dueling Networks



Observation

- DQN with experience replay and fixed Q- target used **2553 episodes** to solve the environment.
- Double DQN with experience replay and fixed Q-target used **1836 episodes** to solve the environment.
- Double DQN + Duelling network with experience replay and fixed Q-target used **2158 episodes** to solve the environment.
- Double DQN + Duelling network could have performed better than Double DQN with some hyperparameter tuning.

Future work

- Hyperparameter tuning using a Grid search and other techniques.
- Prioritized Experience Replay(PER)(link)
- Distributional DQN(link)
- Noisy DQN(link)