# Navigation

Alvaro J. Gaona

December 19, 2021

**Abstract**

First project of the *Deep Reinforcement Learning Engineer Nanodegree* from Udacity. The goal is to start working with Reinforcement Learning algorithms to train any sort of agents. In particular, an agent that collects yellow bananas and avoids blue bananas was trained.

## 1 Introduction

The goal of this project is to train an agent that can detect yellow bananas and collect them, whilst ignoring the blue bananas. The environment is a Unity Environment that comprises of a discrete set of actions and a continuous set of states. The former is a set $\mathcal{A} = \{0, 1, 2, 3\}$ where each value corresponds to up, down, left, right, respectively. The latter, is a continuous set $\mathcal{S}$ of raycast-based perception, and velocity.

## 2 Deep Q-Learning (DQN)

First, an agent that does not rely on the state of the environment and the reward, is implemented to show how bad it behaves. In Fig. 1 the score is plotted throughout all the episodes.

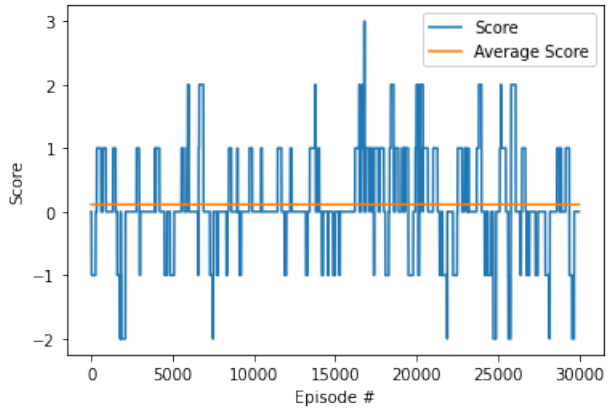


Figure 1: Score per each episode obtained by an agent that selects actions randomly.

Next, a neural network is used to estimate the action-value function $q$. The network consists of a 3-layer fully-connected neural network with ReLU6 activation functions. In additon, the *Mean Squared Error* (MSE) loss function is used.

### 2.1 Reward function

The reward function is fairly simple; Eq. 1 formulates it. In other words, if the banana is yellow and it is collected, then the reward is +1. Otherwise, it is -1.

$$r = \begin{cases} +1 & \text{if banana is yellow} \\ -1 & \text{else} \end{cases} \tag{1}$$

The hyperparameters used are:

- `N_EPISODES = 2000`: Number of episodes to train the agent.
- `MAX_TIMESTEPS = 1000`: Maximum timestemp per each episode.
- `EPSILON_DECAY = 0.99`: Epsilon decay value
- `EPSILON_MIN = 0.001`: Minimum epsilon value.
- `EPSILON_START = 1`: Epsilon start value.
- `LEARNING_RATE = 0.0005`: Learning rate.
- `TAU = 0.05`: Parameter to update the network weights.
- `GAMMA = 0.99`: Discount factor.
- `BATCH_SIZE = 64`: Minibatch size.
- `BUFFER_SIZE = 1000`: Size of the replay memory.
- `GOAL = 14`: Reward value to achieve in training.

Fig. 2 shows the reward for each episode, as well as the average score achieve by the policy.

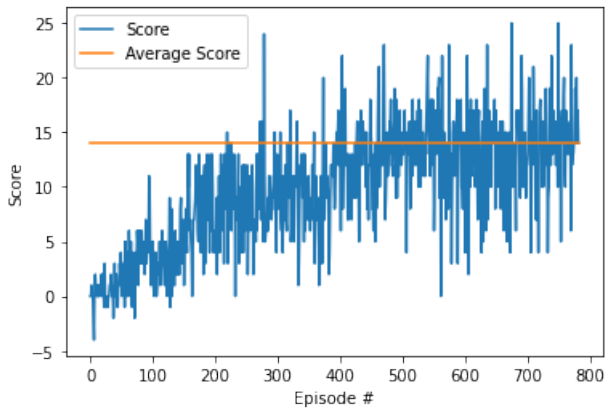


Figure 2: Score per each episode the DQN agent obtains by selecting actions using a neural network.

### 2.2 Improvements

The following techniques to improve training are:

- Experience Replay.
- Epsilon-greedy.
- Fixed Q-Targets.
- Fixed Learning Rate.

## 3 Discussion & Future work

In this project, an agent capable of collecting bananas was trained using DQN techniques. An activation function ReLU6 was used as it had been shown it outperforms other activation functions for this type of tasks; although, the activation comparison was not done for this project.

In addition to the techniques previously mentioned, Double DQN and Dueling DQN can be implemented to improve the training.

More advanced network architectures could be used to fusion pixel information with raycast and odometry information. Moreover, a varying learning rate is possible to apply, as most of the applications using neural networks apply.