

# Deep Reinforcement Learning Specialization – Project 1: Navigation

Vlad Negreanu

## 1 Summary

The project consists in training an agent for self-learning. The agent needs to collect the “good type” of bananas marked as yellow and avoid the “bad type” of bananas marked as blue. The goal is to maximize the yellow collected bananas in the given number of episodes (default is 1000).

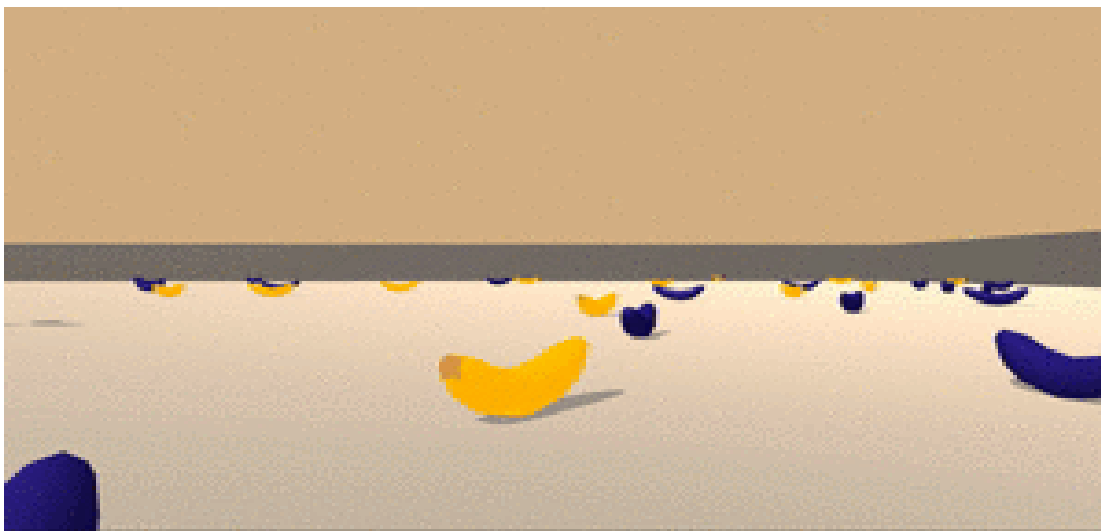
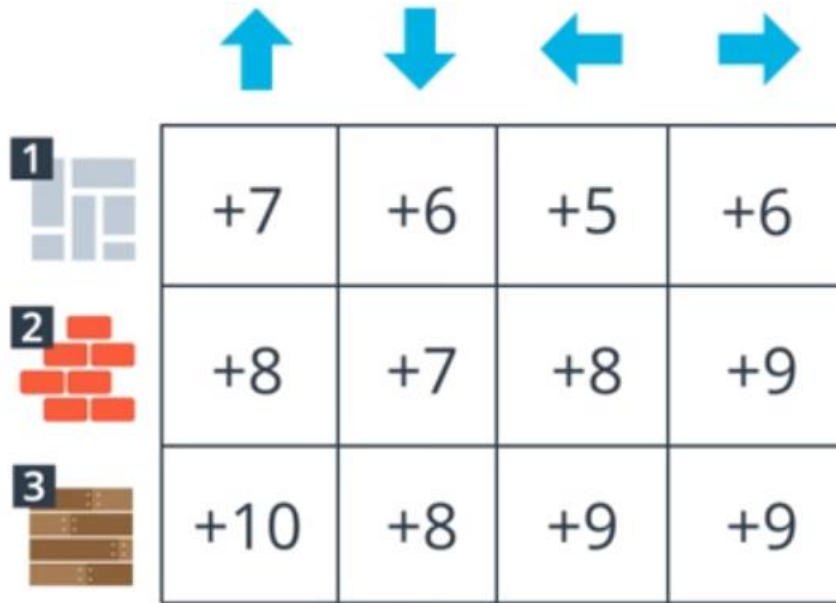


Fig.1 Environment overview

## 2 Reinforcement learning approach

### 2.1 Algorithm decision and neural network description

Method used is **Temporal-Difference (TD)** control method where the estimates are updated based on other learned estimates, without waiting for the final result. TD methods will update the Q-table every time step. Q-table is used to approximate the action-value function  $q$  for policy  $\pi$ . An example is described in the picture below.



1	+7	+6	+5	+6
2	+8	+7	+8	+9
3	+10	+8	+9	+9

Fig.2 Q-table example

The **Sarsamax (Q-Learning)** is one **TD** methods that approximate the optimal value function every time step. Brief description of the algorithm is given by the equation below:

$$Q(St, At) \leftarrow Q(St, At) + \alpha (R_{t+1} + \gamma \max_{a \in A} Q(St+1, a) - Q(St, At))$$

For creating such approximation we will usage **neural network** as follow: **37 cotinuous states** will be marked as the **input layer**, **4 discrete actions** will be marked as **output layer**. And for an optimal value function a **fully-connected neural network** is chosen with the archtecture described below:

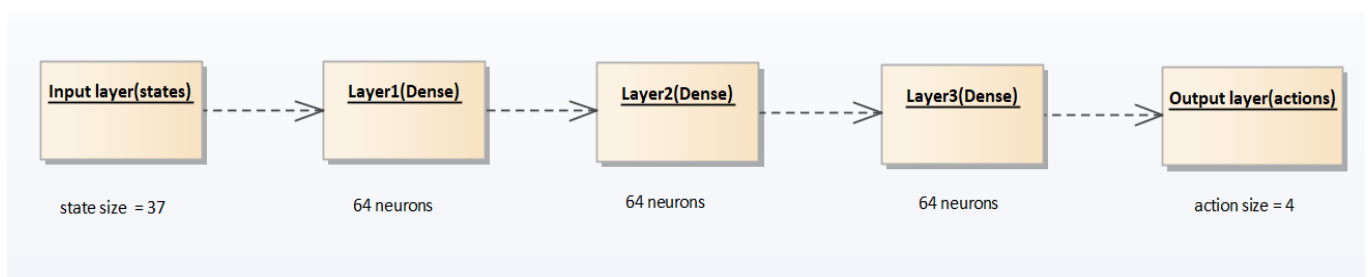


Fig.3 Neural network description

## 2.2 Stabilization issues of the neural networks

Since neural networks are considered unstable when representing action values two features are to be used to reduce the instabilities:

- Experience Replay
- Fixed Q-targets

### 2.2.1 Experience Replay

To interact and learn from the environment a sequence of tuples needs to be stored. For this we keep track of a so called **replay buffer** that contains a collection of tuples  $(S, A, R, S')$  gradually added to the buffer as we interact with the environment. Buffer size used is **50000** so we can store 50000 experiences at a time.

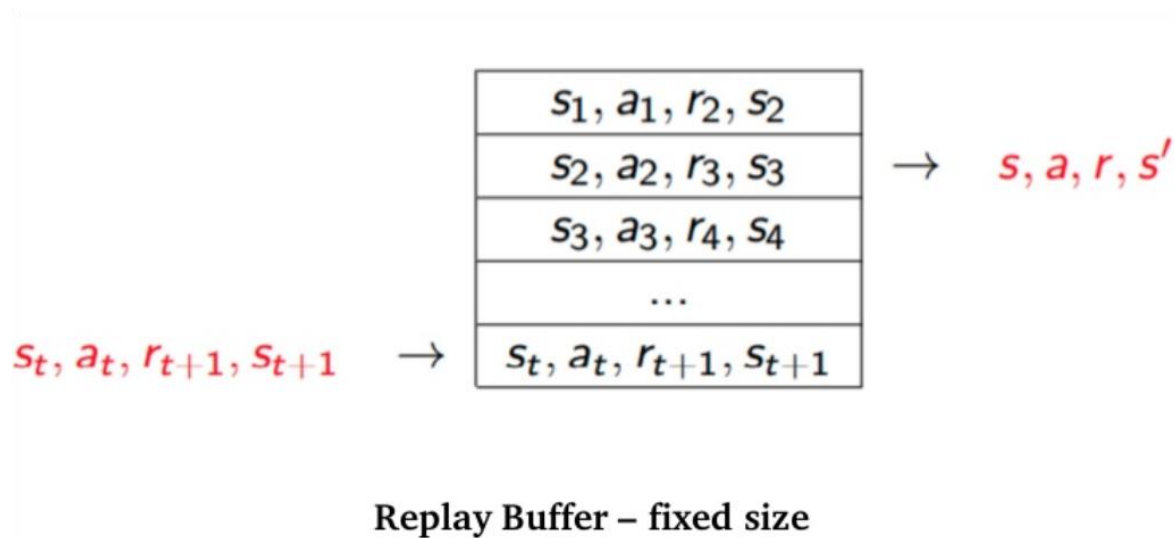


Fig.4 Replay buffer with fixed size

### 2.2.2 Fixed Q-targets

In Q-Learning, when we want to compute the TD error, we compute the difference between the TD target and the current predicted Q-value (estimation of Q) .

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

TD error

Fig.5 Update rule of Q-Learning

But in every step of the training the Q-values shift but also the target value shifts. To overcome this we will use a separate network that has the same architecture Deep Q-Network.

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

↓  
fixed

Fig.6 Update rule of the Deep Q-Learning

In the code we update the target Q-Network weights every 4 time steps.

### 2.3 Other algorithms and parameters

- **Adam** optimizer algorithm since it uses the best properties of the AdaGrad and RMSProp algorithms to provide an **optimization** algorithm that can handle sparse gradients on noisy problems.
- **Learning rate**  $\alpha$  initialized to 5e-4 to avoid overfitting.
- **Discount rate**  $\gamma$  initialized to 0.99 very close to 1 meaning that the return objective takes future rewards into account more strongly as the agent progresses through the environment.
- **GLIE learning policy** which stands for Greedy in the limit with infinite exploration. This strategy first favors exploration vs exploitation and in the final steps viceversa.

### 3 Result

The agent is attempted to be trained initially over 1000 episodes with a computation of cumulative reward per episode. We consider the environment to be solved when the average over the last 100 episodes is greater or equal than 13. In the figure below there is the plot of the score obtained per episodes.

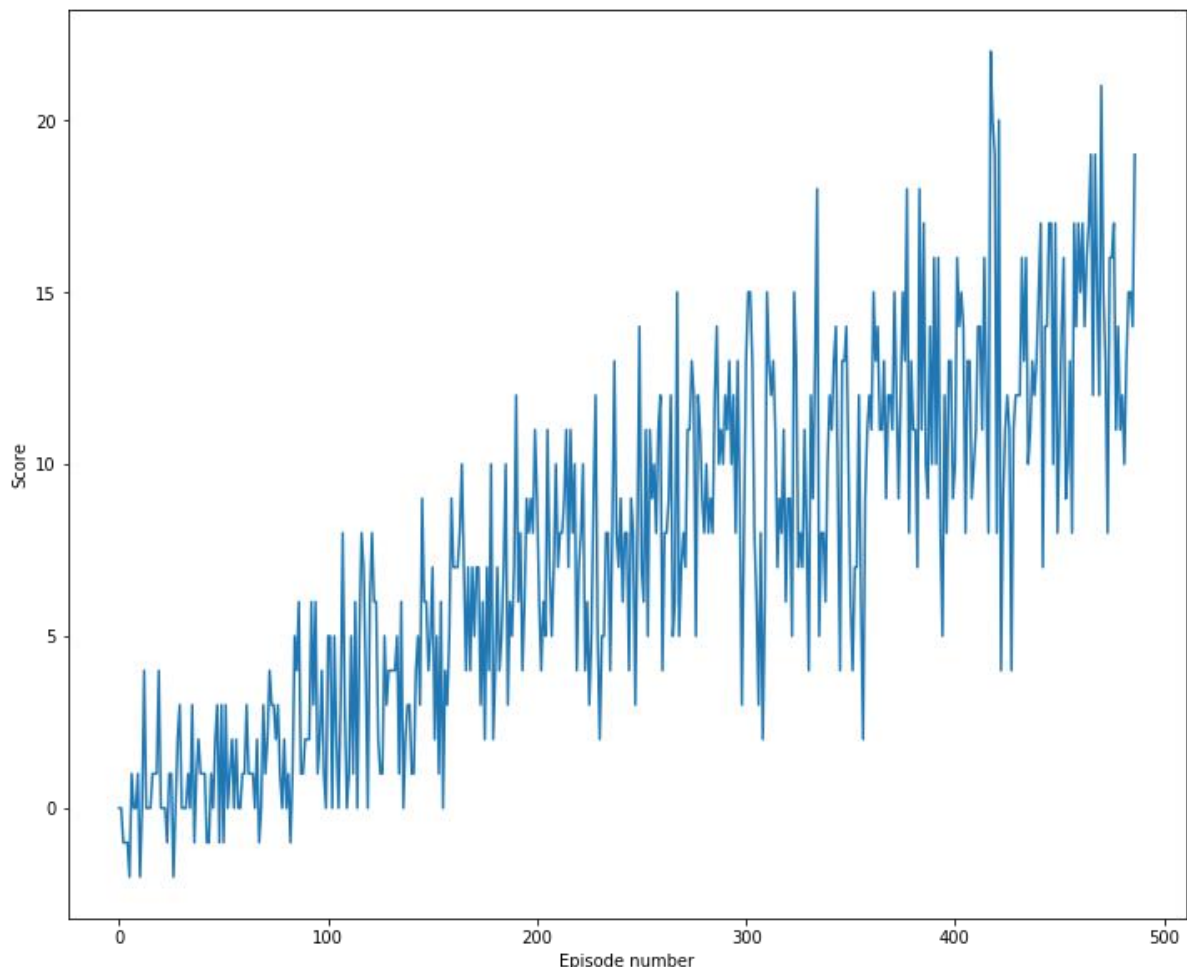


Fig.7 Rewards per episode

### 4 Improvements

- **architectural improvement** can be using a **Dueling network architecture** which leads to improved performance since it decouples the value from the advantage function.

- **stabilization improvement** the main idea is to give much more perspectives to the experiences procedure since it is important for the learning. A possible improvement can be **Prioritized experience replay**.