EL2805 — Reinforcement Learning

Name: João Almeida & Victor Sanchez
Student Number: 19990501-T210 & 19980429-T517
Due Date: December 21 2021, 23:59
Assignment: 2

Problem 1 - Deep Q-Networks (DQN)

Introduction

We will focus here on the discrete action-space version of the Lunar Lander problem. We don't have any information on the model and the state space is continuous. We have a finite action space so we will using Deep-Q-Networks to solve make the lunar landing possible.

Question (a) Problem formulation as an MDP

A MDP consists is composed by a State Space (S), an Action Space (A), Transition probabilities (T) and a Reward (R) $\tilde{M}$(S,A,T,R).

State Space: It is an 8-dimensional variable

$$S = \{s_1\ s_2\ s_3\ s_4\ s_5\ s_6\ s_7\ s_8\}$$

s_1 : Position on x-axis s_2 : Position on y-axis s_3 : Velocity on x-axis
s_4 : Velocity on y-axis s_5 : Lander angle s_6 : Angular velocity
s_7 : Left contact point s_8 : Right contact point

Action Space: Discrete Action Space :

$$A = \{A_1\ A_2\ A_3\ A_4\}$$

A_1 : do nothing ; A_2 : fire left orientation engine
A_3 : fire main engine ; A_4 : right orientation engine

Transition: All the transition probability are contained in the environment.

Rewards: r(lander crash) = -100 ; r(come to rest) = +100
r(leg ground contact) = +10 ; r(fire main engine) = -0.3 ; r(fire side engine) = -0.03

Questions (b)

In Reinforcement learning algorithms, we use a replay buffers to store experience trajectories when executing a policy in an environment. They are really practical because during training, the replay buffers are polled for a subset of the trajectories (either a sequential subset or a sample) to "replay" the agent experience.
In addition, we use a target network in DQN to use it as a memory of the past main network and thus to update the new main network for the next experiment.

Questions (c)

We have implemented the replay buffer with the same as we did in the lab0.
First we fill it at 20% of its capacity with random action and state. During the learning the buffer is growing with the results of each experiment.
The neural network we choose, has two hidden layers with 64 neurons each. The activation function are ReLu everywhere. The choice of 64 neurons in both hidden layers will be seen later on the report. We decided to used the optimizer Adam because we follow the tips and tricks part.

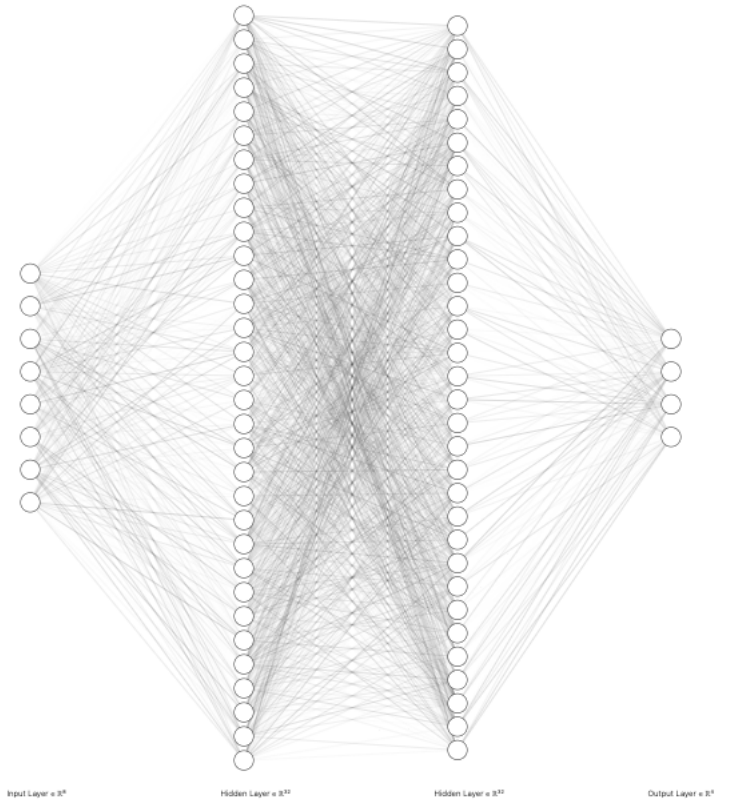

Figure 1: Drawing of the neural network

Questions (d)

Regulation of the discount factor : γ

The discount factor essentially determines how much the reinforcement learning agents cares about rewards in the distant future relative to those in the immediate future. If $\gamma = 0$, the agent will be completely myopic and only learn about actions that produce an immediate reward.

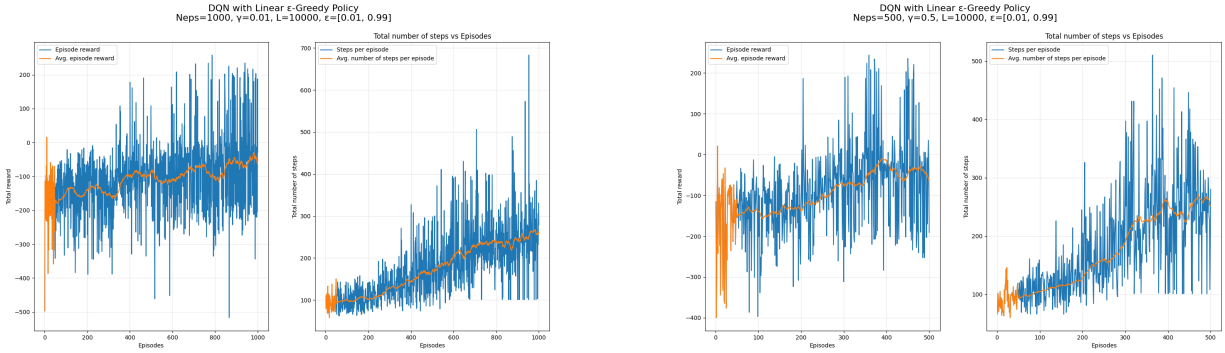

Figure 2: DQN with exponential ϵ-greedy with γ=0.01 and γ=0.5

We first start with a low value of discount factor and we can observe that the learning process is so super slow, after 500 steps the average is still negative.

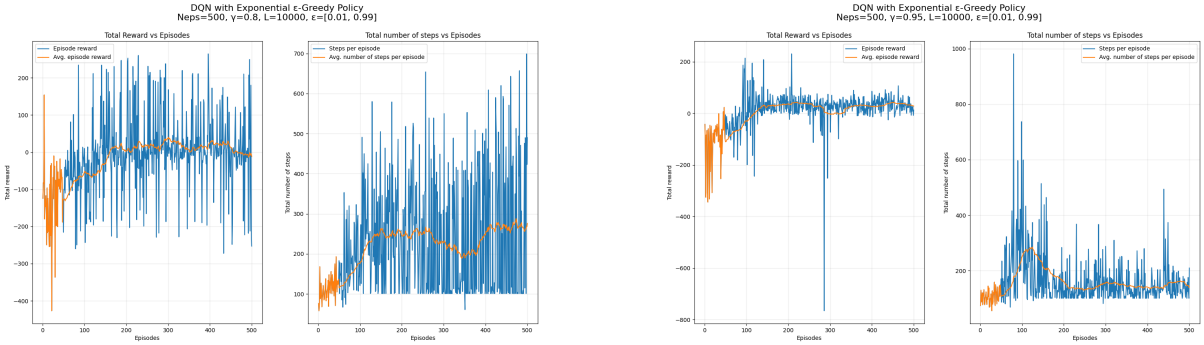

Figure 3: DQN with exponential ϵ-greedy with γ=0.8 and γ=0.95

As we increased the discount factor until 0.95 we observed that the learning is improved but still not learning efficiently enough.
On the other hand, for γ=1 the agent evaluates each of its actions based on the total sum of all of its future rewards. So after a certain number of episodes, the learning is not efficient and the average reward start to decrease a lot.
However for γ=0.99, we can see that the agent is learning much better compared to any of the other cases. So for the rest of the lab, we will keep that value.

Choice of the buffer size : L

We start to use the values proposed on the tips and tricks part. We can observe that the extreme value proposed are both working well in terms of learning.

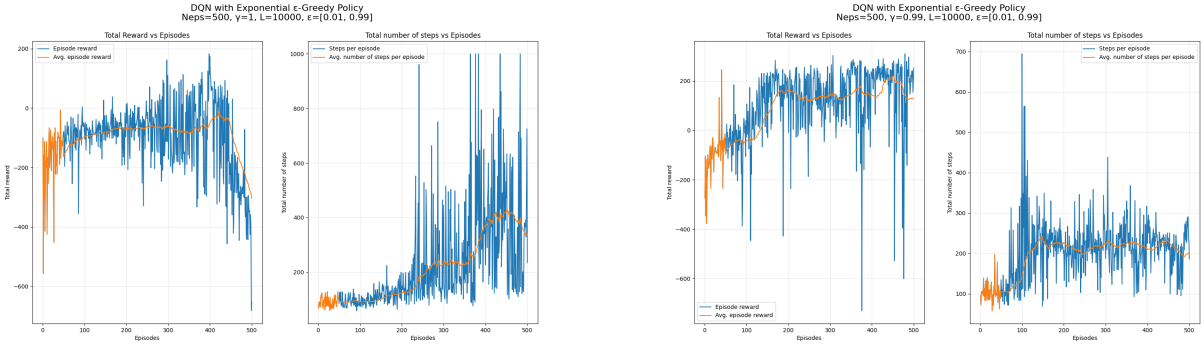

Figure 4: DQN with exponential ϵ-greedy with γ=1 and γ=0.99

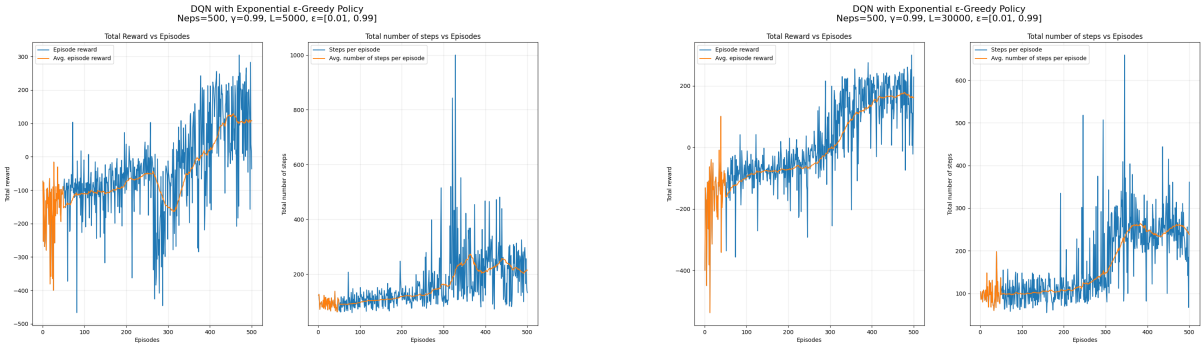

Figure 5: Comparison of DQN resolution with respective buffer size 5000 and 30000

Based on the graphs above, both overcome an average reward of +50 in the last the prior 50 episodes at the end, even though the higher value of L allowed to have a better results but also more steps specially in the last third of the episodes. Therefore it was increased the value of relay buffer to 10000 trading a computational cost to a better result. The group could keep increasing the value, but was satisfied with the results obtained.

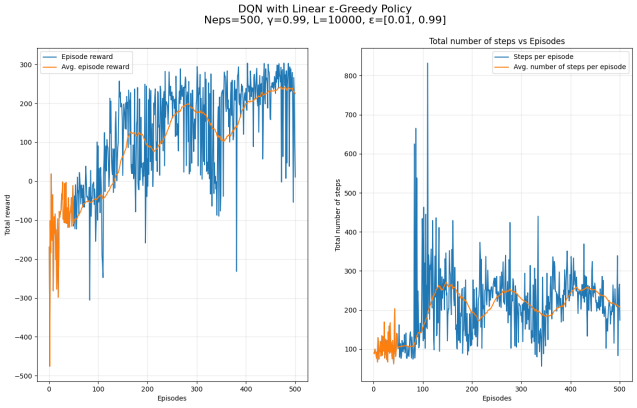

Figure 6: Result of DQN resolution with buffer size of 10000

Choice of the number of episode : T_E

The number of episode has it importance because if it is not large enough, the agent won't be able to learn properly.
We can see that phenomenon when $T_E = 250$ and when $T_E = 100$.

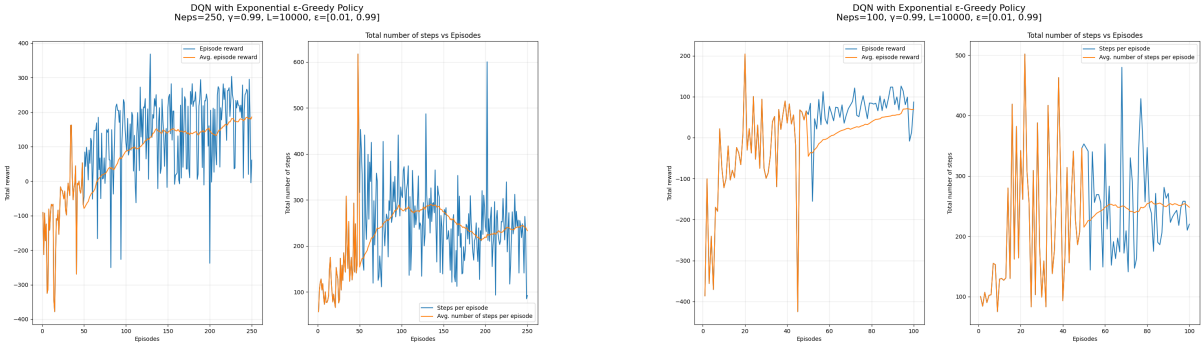

Figure 7: Comparison of DQN resolution with respective number of episode : 250 and 100

Nonetheless, a large number of episode is much more practical to learn better the experiences. But the trade off that the computational time increase with the number of episode.

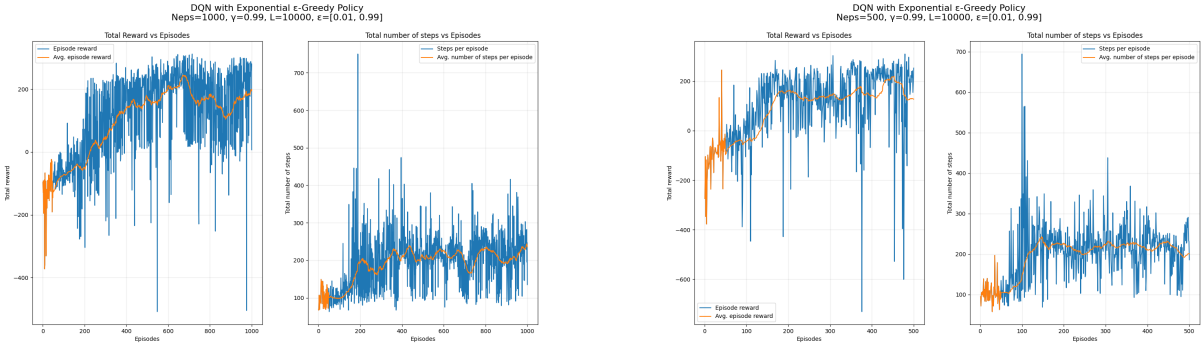

Figure 8: Comparison of DQN resolution with respective number of episode : 1000 and 500

To not reach the maximum number of episode and reduce the training time we choose $T_E = 500$ episodes.

Choice of the training batch size : N

During the whole training process we took either 32 or 64 as size for the batch. We noticed also that the batch size has to be more or less proportional to the discount factor. As we kept a discount factor equal to 0.99, we decided to use a batch size of 64.

Choice of the target network frequency update : C

We have chosen the frequency update with the C = L/N.

Choice of the decay parameter ϵ

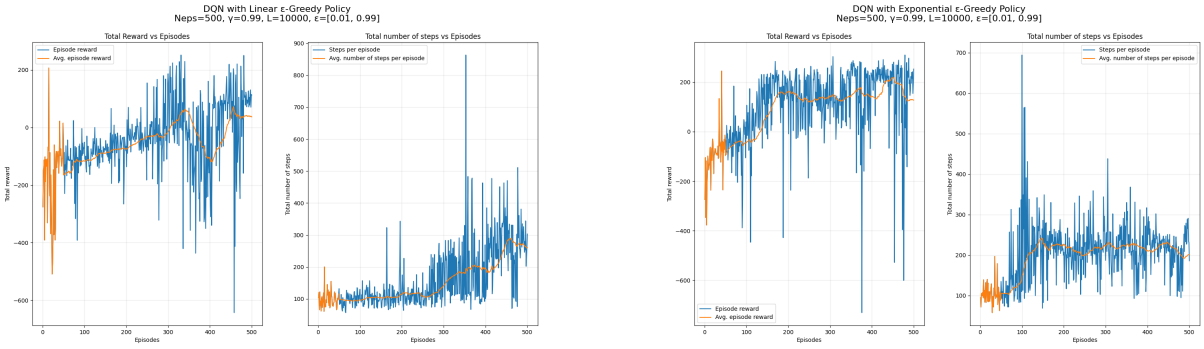

Figure 9: Comparison of DQN resolution with linear and exponential decay parameter

It is also possible to identify the how the decay factor is decreasing. First the range of the value is [0.01, 0.99]. In left graph of the figure 9, it is seen an linear increasing of the reward value, and on the right side, the learning process has an exponential behaviour.
With all the parameter we selected before we are now comparing the exponential and the linear mode for the decay value.
We can interpret from these results that the agent is learning faster and more efficiently with the exponential model.

Modification of implementation during training

During the learning process, an impressive number of episodes with 1000 steps, which were consuming much time in the simulation. In order to overcome, the group implement an otimization that says: If the average speed in the y axis is above a certain threshold the agent is not learning. This applies not only when the agent is stuck in the ground, hover or already landed but somehow does not turns of the motors. Regarding the hypothesis,and since the flag coordination are not known, when the y coordinate is almost 0, the agent is awarded +50 that symbolizes that he landed in the correct place but not in the correct way (reward = +100).
We can notice from the plots in the left part that the learning is efficient but it is actually taking a long time to process and reach the total number of episode a lot of time. Thus, it is taking a lot of time to train.
In the right part, the total number of episode is reaching the threshold of 1000 only on a few experience which means that the training is much faster.

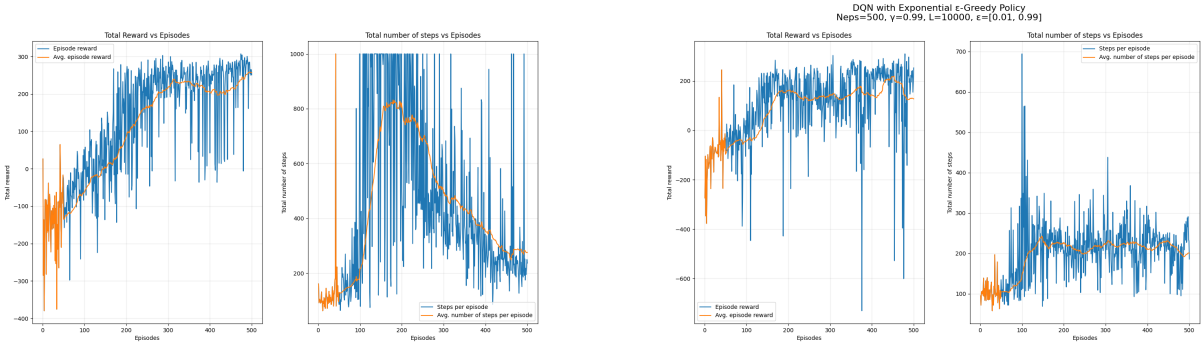

Figure 10: Comparison of DQN resolution before (left part) and after (right part) improvement of learning

Summary of parameter choosing

γ	L	T_E	N	C	ϵ
0.99	1e4	500	64	$\frac{L}{N}$=156	exponential mode

Questions (e)

(1) - Training process

Plot of episodic reward and the total number of steps taken per episode during training.

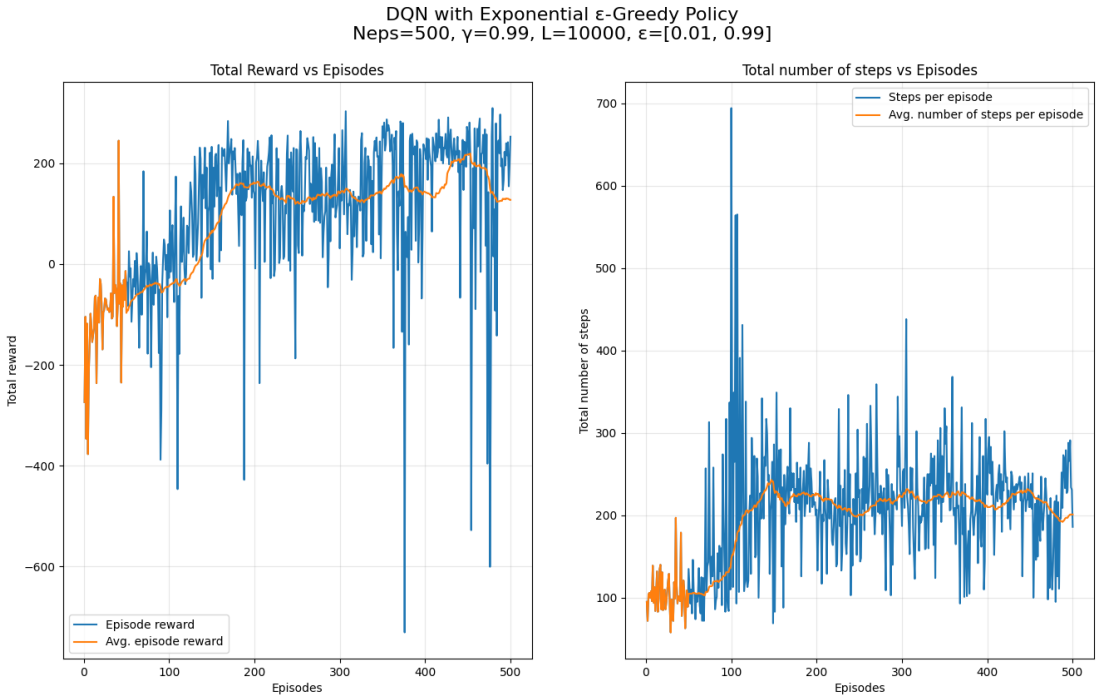

Figure 11: DQN resolution result

The training process is efficient in terms of time because it is taking a total average per episode of around 200. And also the total reward is positive and higher than 50, so we can conclude that our algorithm solves the problem.

(2) - Discount factor chosen : $\gamma_0 = 0,99$, $\gamma_1 = 1$, $\gamma_2 = 0.01$

As we have seen before, if the discount factor is low, the agent will be completely myopic and only learn about actions that produce an immediate reward. Then the training process is going to be distorted and it will be very slow. On the contrary, when the discount factor is equal to one, the agent take decision only based on the ones of its future and not it's past.
We are choosing 0.99 to be as close as possible from that without falling into the curse.

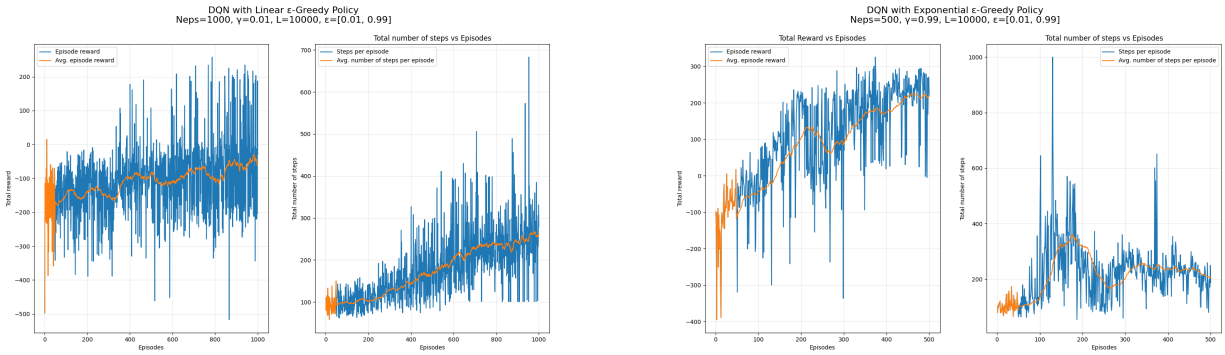

Figure 12: DQN with exponential ϵ-greedy with γ=0.01 and γ=0.99

(3) - Influence of number of episodes and memory size

The effect of the number of episode is twice. First, if we increase it, the computing time increases also. But the learning process is much more efficient because the agent learns during more episodes. Secondly, we spend more in in each episode because the decay process is slower so the all process is slower than before.

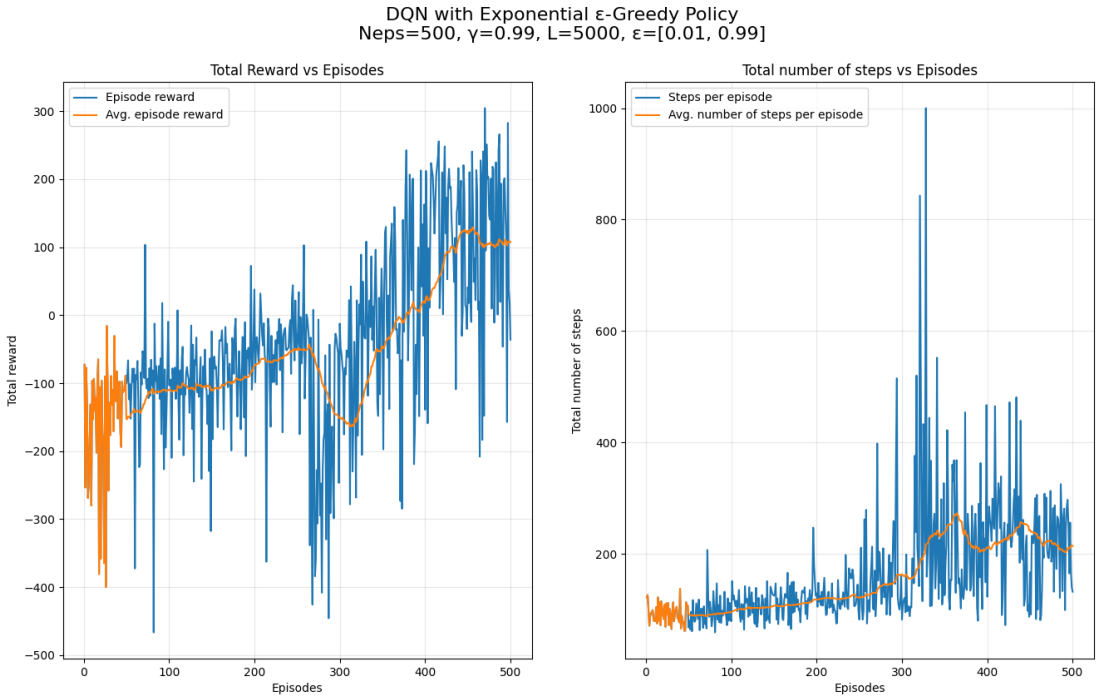

Figure 13: DQN Resolution with memory size of 5000

Questions (f)

We consider s(y,ω) = (0,y,0,0,ω,0,0,0).
We plot here the $max_a Q_\theta(s(y,\omega),a)$, $argmax_a Q_\theta(s(y,\omega),a)$ for $y \in [0, 1.5]$ and $omega \in [-\pi, \pi]$:

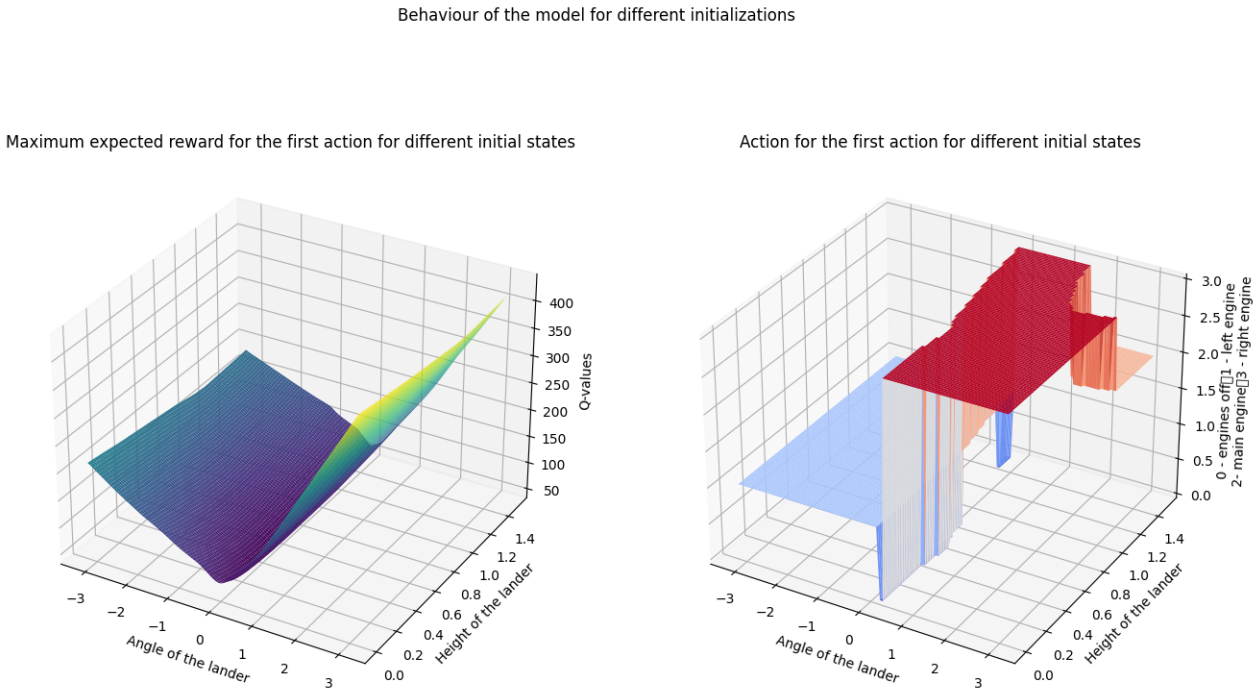

Figure 14: DQN module for a given restricted state

When the angle of the lander is zero, the Qvalue is +50, which is due to the otimization explained here .
The graph itself does not make much sense, since when the angle is 0, it does not receives the same rewards it might receive when the initial angle differs from 0, although the group thinks that the environment chooses that based because there is no need to have the engines working and so in a real case scenario would make the aircraft unstable and not unable to react to external stimulus. It is also unclear that the higher and bent the aircraft is, the more rewards it gets, but we think if we trained with more episodes, the shape of the graph would be different. Theses imperfections can be seen in the prior plots, when in the last states when the average reward is positive there are still episodes with bad rewards, being this extreme cases.
Regarding the right plot, it is possible to see almost a perfect boundary when the angle is 0, symbolizing when the motors are turned off. Then when the angle is negative the agent chooses to use the left motor and when is positive the right motor in other to straighten up the aircraft.

Questions (g)

Finally we compare the Q-network we have obtained with the random agent decision taking we had before over 50 episodes only. We will first observe the average reward and then we will reproduce the scenario of the former question f).

Average reward viewpoint

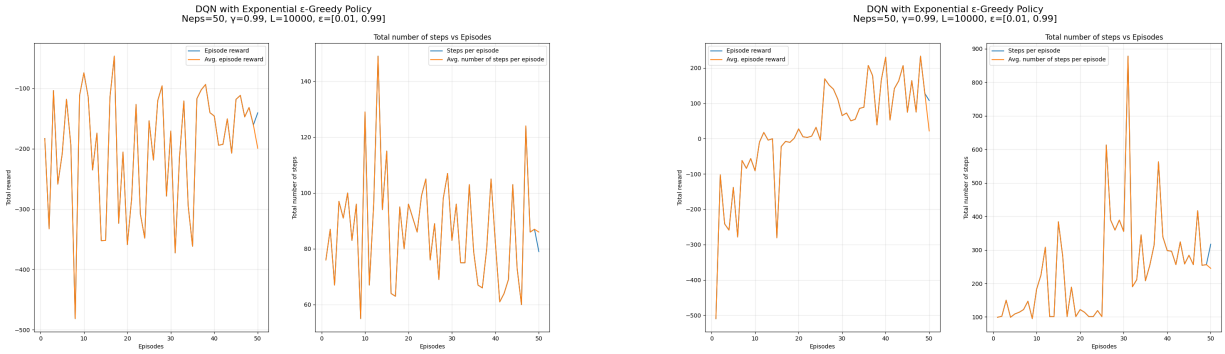

Figure 15: Comparison of DQN resolution with Random agent and Improved agent :

We can observe finally clearly here that the random agent is not learning because the reward does not improve and stays negative. The improved agent is on the contrary much better because it reaches a very good average reward.