EL2805 Reinforcement Learning Lab 2

Claire Binet-Tarbé de Vauxclairs (20020701-T842), Marceau Messaussier (20030805-T390)

30 December 2025

Problem 1

a.

The random agent achieves a total reward of -180 ± 3 over 5000 episodes. To implement it, we just take a random action for each action. We will use this result as a comparison baseline for our models.

b.

We use a replay buffer to prevent experiences from being too much correlated. We also use it in order to prevent the 'forget' issue. With it our model can learn from previous experiences several times. The target network, on the other hand, is the network used to stabilize the training, we need it because otherwise we would modify the network while learning. We want a more "stationary" target.

d.

The optimization process was conducted using the **Adam** optimizer, selected for its robust performance in deep reinforcement learning tasks and its ability to adaptively tune learning rates. We set the learning rate to 10^{-4}; this relatively conservative value was chosen to ensure stable convergence and to facilitate the monitoring of incremental performance improvements during the initial debugging and training phases.

The baseline configuration was defined by the following set of hyperparameters:

$$\gamma = 0.98, \quad L = 20,000, \quad T_E = 800, \quad C = 312, \quad N = 64, \quad \varepsilon_k = \max(0.999 \cdot 0.99998^k, 0.05) \quad (1)$$

The selection of these parameters was guided by both theoretical heuristics and empirical observations:

- **Discount Factor (γ):** We chose $\gamma = 0.98$ to ensure the agent accounts for long-term rewards while remaining strictly less than unity to ensure Q-value convergence. This value also encourages slightly more efficient trajectories (shorter episodes) compared to a higher γ like 0.99.
- **Replay Buffer Size (L):** A capacity of $20,000$ transitions was implemented to manage the high state-space complexity of the environment. This large buffer assists in decorrelating the training data by providing a diverse history of trajectories.
- **Training Duration (T_E):** While the environment was technically "solved" (reward > 200) around episode 600, training was extended to 800 episodes. This additional duration allows the agent to move beyond mere survival and further optimize its control policy to reduce the total number of steps.
- **Target Network Update (C):** The update frequency was set to $C = 312$, following the general heuristic of $C \approx L/N$ to maintain a stable learning target.
- **Exploration Strategy (ε):** The epsilon-decay schedule was calibrated to the total expected number of steps. This ensures a smooth transition from exploration to exploitation, reaching a terminal exploration rate of $\varepsilon \approx 0.05$.

Regarding the architecture, our initial implementation utilized a standard three-layer fully connected network. However, we observed significant volatility and performance degradation after approximately 500 episodes. To mitigate this variance and stabilize the learning process, we integrated **Combined Experience Replay (CER)** and transitioned to a **Dueling Network** architecture. The final dueling model structure is detailed below in Figure 1.

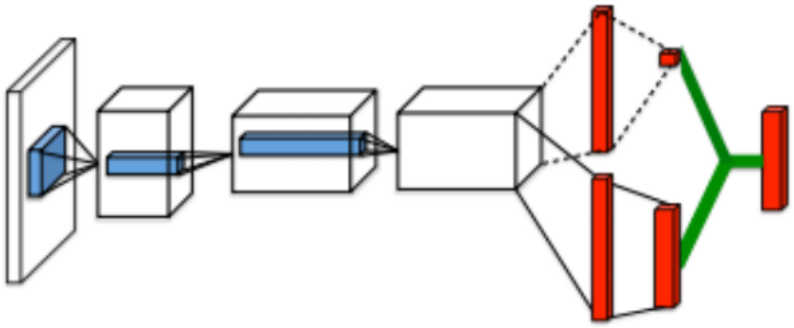

Figure 1: Dueling structure of the network. We implemented one input layer connected to two different streams, the value stream and the advantage stream. Value stream is composed of two layers, the first has 128 neurons then followed by a single neuron it's gonna represent the value $V(s)$ predicted by the network. The other stream is the advantage one, it is composed of two layers, the first one has 128 neurons and the second one has 4 neurons (to represents the advantage values for each one of the 4 actions).

e.

(1)

During the training process, the average reward per episode shows an overall increasing trend, although it does not follow a perfectly monotonic path. In the early stages of training, there is no clear performance tendency; the agent is likely exploring the state space and frequently crashing, leading to low and unstable rewards. A major breakthrough occurs around episode 300, where the total reward begins to increase significantly. Another noticeable 'step' or jump in performance at approximately 550 episodes is also observed. By episode 600, the learning curve flattens out, and no further significant gains are observed, suggesting the model has converged to a stable policy.

Regarding the number of steps per episode, before episode 300, the step count is relatively low because the agent crashes almost immediately. However, right at the 300-episode mark—coinciding with the first major reward increase—there is a sharp spike in the number of steps. This indicates that the agent has learned to 'survive' and stay in the air longer, even if it hasn't mastered the landing yet. After this peak, the number of steps starts to slowly decrease as the agent becomes more efficient. It finally reaches an equilibrium at 600 episodes, stabilizing at approximately 400 steps per episode. This transition from a high step count to a lower, stable one shows that the agent is now optimizing its path to the landing pad. These trends and correlations are illustrated in Figure 2.

(2)

Note: It should be noted that the baseline data for questions (e.2) and (e.3) comes from a separate training session than the one used in (e.1). Because results vary slightly between runs, the baseline curves shown hereafter may differ slightly from the initial training plots.

Training results for different discount factors are presented in Figure 3. The Baseline ($\gamma_0 = 0.98$) appears to be the most effective, reaching a high stable reward of over 200 by episode 400. Interestingly, $\gamma_1 = 1$ (which weights all future rewards equally with current ones) follows a similar learning trajectory but shows a higher spike in the number of steps and slightly more oscillation after the peak has passed. This suggests that while a high γ allows the agent to solve the task, a value of exactly 1.0 might make

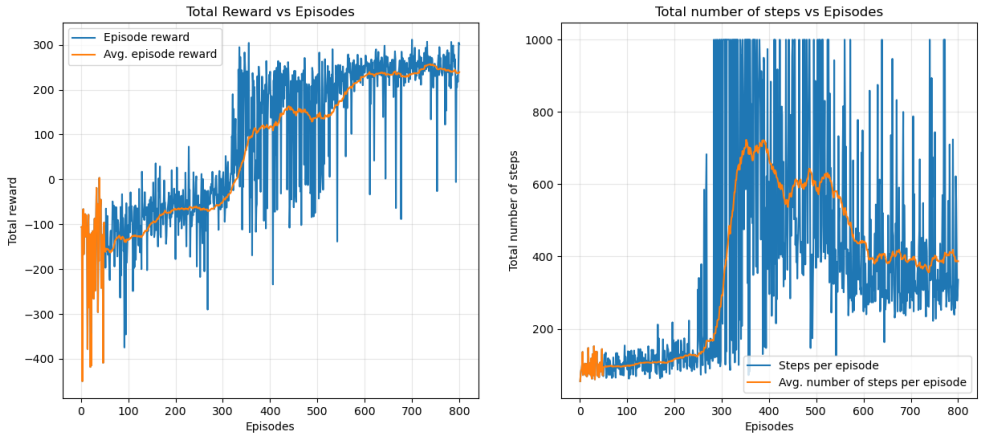

Figure 2: Total rewards per episodes (right) and total numer of steps per episodes (left) with configuration parameters of question d

the Q-values harder to converge because the agent is trying to account for an infinite horizon of future rewards without any decay.

On the other hand, the low discount factor ($\gamma_2 = 0.5$) performed significantly worse. As seen in the Average Reward graph, the agent fails to ever reach the success threshold, plateauing near a reward of 0. This behavior occurs because a low γ makes the agent prioritize immediate rewards too heavily, failing to understand that certain early maneuvers are necessary for a successful landing much later in the episode. This is further supported by the Average Steps graph, where the $\gamma_2 = 0.5$ case shows a much lower peak in step count; the agent never truly learns the 'survival' phase and likely crashes or ends episodes early because it lacks a long-term strategic perspective. In conclusion, a γ close to 1 is essential for the Lunar Lander task to ensure the agent values the final successful landing as much as the initial flight adjustments."

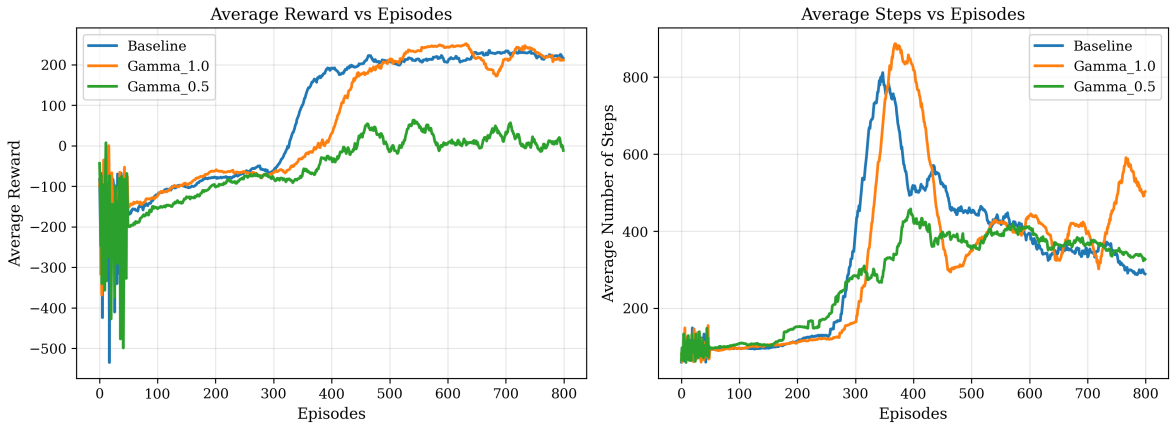

Figure 3: Training caracteristics with different discount factors : 0.98 (basseline), 0.5 and 1

(3)

Number of episodes: By comparing training lengths from 200 to 1,000 episodes, we can identify the minimum threshold required for a robust solution (Figure 4.

- **Short Training (200 episodes):** As seen in the blue curve, 200 episodes are insufficient. The agent is still in the early exploration phase, with rewards stuck at negative values and a step count that hasn't yet reached the 'survival spike.'

- **Medium Training (400 episodes):** At 400 episodes, the agent has just reached the breakthrough point. While it has started to solve the environment, the policy hasn't fully stabilized yet, as evidenced by the high number of steps still present at the end of the run.
- **Long Training (800 and 1,000 episodes):** These durations allow for full convergence. Between 600 and 1,000 episodes, the reward curve reaches a definitive plateau (around 200–250). More importantly, Figure 4 shows that after the initial survival peak at episode 400, the agent uses the additional episodes to optimize its descent, eventually reducing the step count to a stable equilibrium. This confirms that while the 'solution' is found around episode 500, additional training is required to achieve an efficient and reliable landing policy.

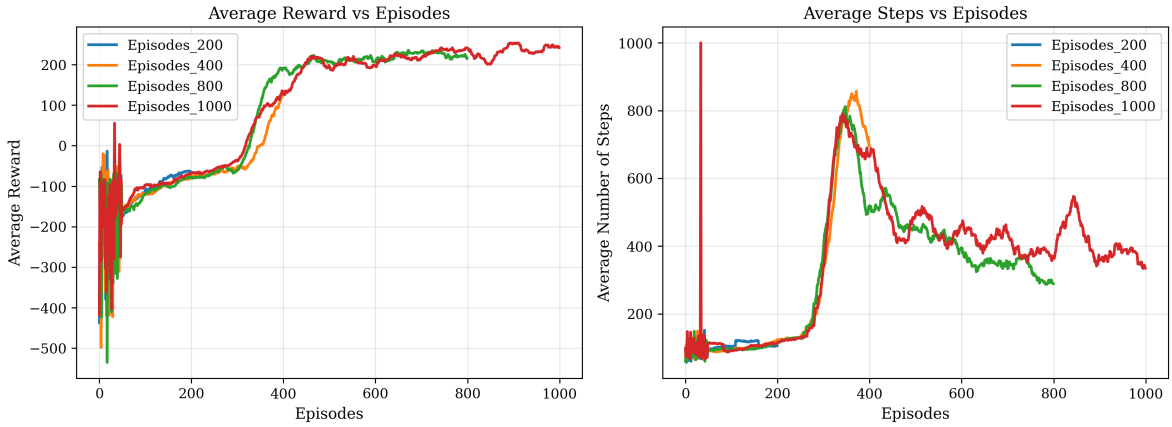

Figure 4: Impact of the number of episodes on the training for 4 parameters different: 200, 400, 800 and 1000 episodes

Replay buffer size: The size of the Replay Buffer proves to be a decisive factor for the stability and success of the DQN agent, as shown in the comparison between sizes ranging from 500 to 50,000 (Figure 5).

- **Small Buffers (500 and 1,000):** These settings generally result in highly unstable learning trajectories. With a buffer of only 500, the reward curve exhibits significant oscillations and fails to sustain a high performance level, eventually regressing after an initial rise. This instability occurs because the agent 'forgets' older, diverse experiences too rapidly, leading to overfitting on its most recent—and often unsuccessful—attempts. Consequently, the Average Steps plot for these smaller buffers shows erratic spikes, indicating that the agent struggles to develop a consistent landing strategy. Interestingly, while the 1,000-element buffer eventually achieves a successful landing with the lowest step count across all simulations, this result should be interpreted with caution; such performance may lack consistency or reproducibility across different training sessions due to the limited diversity of the data in the replay memory.
- **Large Buffers (20,000 and 50,000):** These sizes provide the best results. The Buffer 50000 curve (red) is the most stable, reaching a high reward plateau earlier and maintaining it with minimal variance. A larger memory allows the agent to sample from a diverse history of states, preventing it from being misled by a string of recent bad episodes. In the steps plot, we see a clean peak followed by a smooth stabilization, which is a hallmark of an agent transitioning from survival to path optimization.

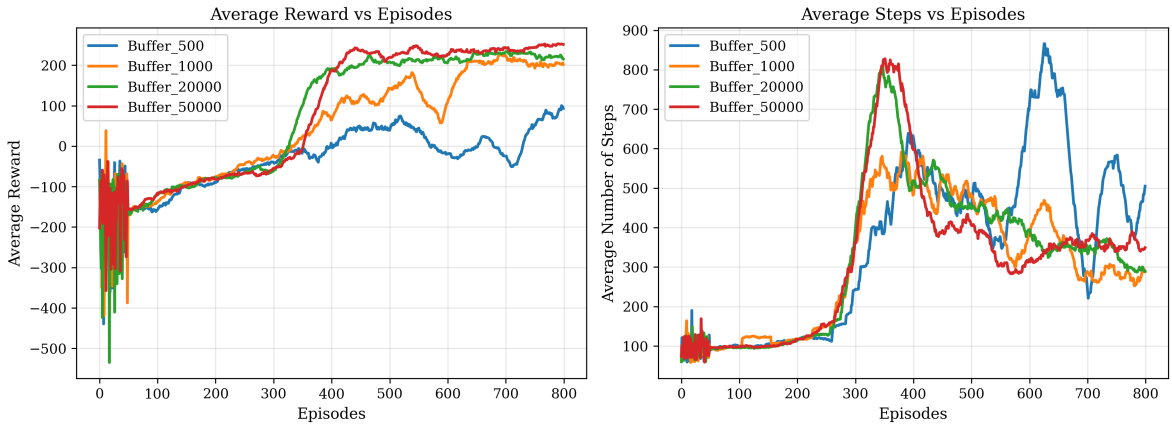

Figure 5: Impact of the memory capacity for 4 different trainings with the following replay buffer sizes: 500, 1000, 20000 (baseline) and 50000.

f.

(1)

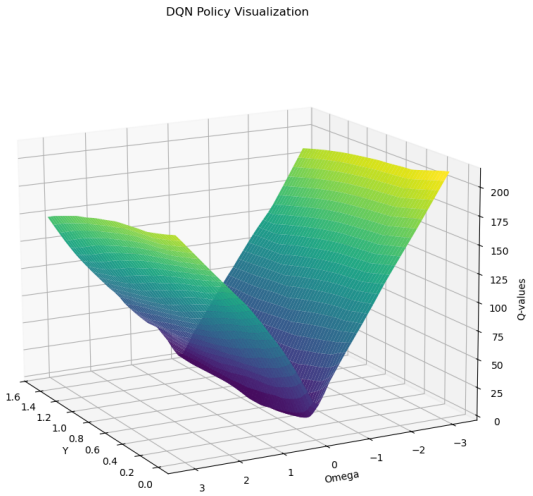

Figure 6: Estimated Q-values of the model. We plot here $\max_a Q(s,a)$, which corresponds to $V^\pi(s)$, for states $s(y,\omega)$ with $y \in [0, 1.5]$ and $\omega \in [-\pi, \pi]$. The obtained shape does not match our physical intuition. One would expect a similar shape but inverted, since states with $\omega = 0$ correspond to ideal configurations where no orientation correction is required and only the main engine should be used. Instead, we observe higher values for extreme angle configurations, while $V(s(y, \omega = 0))$ remains close to zero. Moreover, the values observed at extreme angles are closer to what one would expect around $\omega = 0$, namely $V(s, \omega = 0) \approx 250$ rather than ≈ 0. However, what matters most for control is the action selected by the policy (as discussed in question (2)). This behavior can be explained by the fact that during training, the angle ω remains close to zero in most episodes. As a result, the model learns meaningful Q-values only in this region and extrapolates arbitrarily for extreme angle values, which are rarely or never encountered during training.

(2)

We observe in the following figure that when the angle ω is positive, the model tends to activate the right orientation engine, while for negative values of ω it tends to activate the left orientation engine, which is consistent with the expected behavior. The available actions are:

- 0: do nothing,
- 1: fire left orientation engine,
- 2: fire main engine,
- 3: fire right orientation engine.

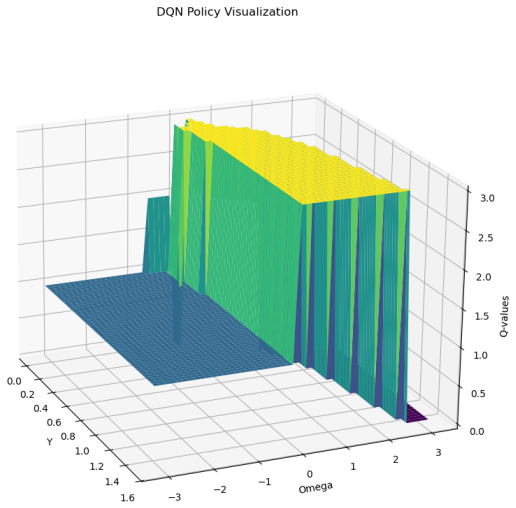

Figure 7: This plot represents the policy $\pi : (y, \omega) \mapsto \arg\max_a Q(s(y, \omega), a)$. The observed behavior is coherent: when the angle is positive, the right orientation engine is activated, and when the angle is negative, the left orientation engine is activated. The following figure illustrates the behavior of the policy when the module is closer to the ground.

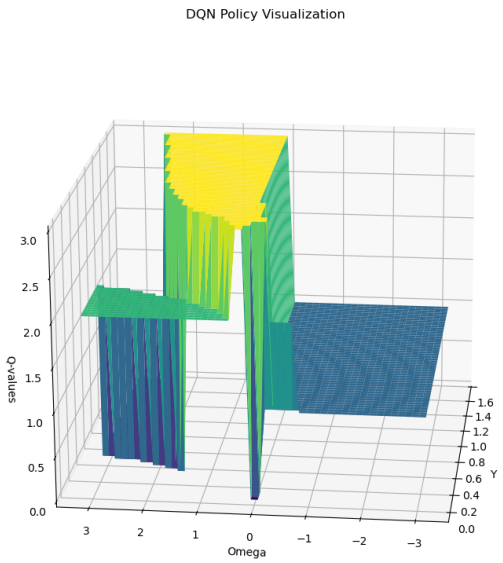

Figure 8: This figure shows the back view of the previous 3D plot. It represents the policy when the module is close to the ground, as a function of the angle ω. We observe that the policy first tends to reorient the module toward a positive angle ω, and then activates the main engine to control the descent. However, incoherent policy decisions can be observed for extreme angle values. For instance, when the angle is close to $+\pi$, the policy activates the main engine even though the module is upside down. As discussed in the previous question, this behavior is probably due to the lack of training samples corresponding to such extreme configurations.

g.

As mentioned in the first question, the random agent is pretty bad and achieves a total reward of -180 ± 3 over 5000 episodes (or -186.4 ± 30.8 over 50 episodes) with 95% confidence interval while our model achieves a total reward of 223 ± 6 over 500 episodes (or 226.0 ± 18.6 over 50 episodes) which is way better.