University of Liverpool
Department of Computer Science

COMP532 - Machine Learning and Bioinspired Optimisation
Lunar Lander: A Deep Learning Reinforcement Approach

Olisa Ajufo [1] Martins Eweniyi [2] Noel Ogbuagu [3] Walid Salem [4] Vasiliki Nikolaidi [5]

April 22, 2024

Lunar Lander, Deep Q Network & Exploration vs Exploitation

Lunar Lander refers to a spacecraft designed to land on the moon, and one of its design requirements involves the appropriate use of propulsion to decelerate and achieve a soft landing [Connolly, 2020]. The Luna Lander environment, which is part of the Box2D environment offered by Open AI Gymnasium, implements a digital model of the real lunar lander, and it can be further customised to add constraints such as gravitational pull, wind, wind power, and turbulence power [Gymnasium, 2022]. This assignment aims to utilise a deep-learning reinforcement approach to train the lunar lander so that it lands between the flags of the environment, which is the target position as shown in Figure 1. These flags do not change positions across episodes.

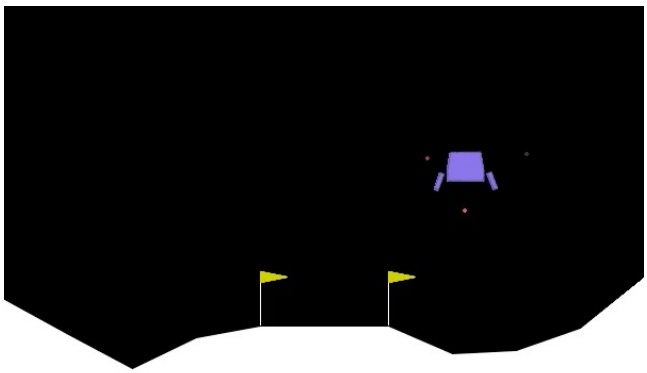


Figure 1: OpenAI Gymnasium Lunar Lander Environment

Our approach uses a Deep Q Network architecture for the deep reinforcement learning process. Although fully connected neural networks are known to be unstable when presented as an action-value function [Katnoria, 2017], this problem has been overcome by using two methods. A memory buffer records the playing experiences and trains the Deep Q network by randomly sampling the episodes. The training process becomes more stable and efficient by updating the Q-network parameter using batches of experiences from the memory. The formula for updating the network parameters is based on the Bellman equation:

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha(r + \gamma \max_{a} Q(s',a'))$$

Where $Q(s,a)$ represents the Q-value for the state-action pair (s,a), α is the learning rate, r is the immediate reward, γ the discount factor, and (s',a') the next state-action pair that maximises the Q-value for that state.

[1]SID: 201763596 [2]SID: 201759964 [3]SID: 201719039 [4]SID: 201733113 [5]SID: 201772172

Furthermore, the fixed network weights are updated while training to optimise the network and achieve a higher score. The epsilon greedy strategy is used in our implementation, and the epsilon decay method is experimented with to balance exploration and exploitation. This method decreases the exploitation rate gradually. The lander chooses between trying random actions and using actions based on previous experiences. Epsilon decay is a dynamic strategy [Dhoble et al., 2020] where the exploration rate is decreased as follows:

$$\epsilon_{t+1} = max(\epsilon_{final}, \epsilon_{initial} \times decay\_factor)$$

Exploration and exploitation are controlled by the epsilon parameter, which starts high to encourage exploration but decreases over time to promote exploitation as the agent learns. Figure 2 shows that different decay rates influence how quickly the agent shifts from exploration to exploitation across different episodes. An appropriate balance allows the agent to explore sufficiently to learn the environment while still exploiting learned strategies for maximum performance. In the Lunar Lander example, while the episodes progress, the agent learns that landing on uneven surfaces or flying over the borders of the environment results in a penalty; thus, the moves are avoided in later episodes. The epsilon decay rate is crucial in achieving this balance. The choice of decay rate can impact the agent's learning efficiency and outcomes. After experimentation, a decay rate of 0.991 was selected for training.

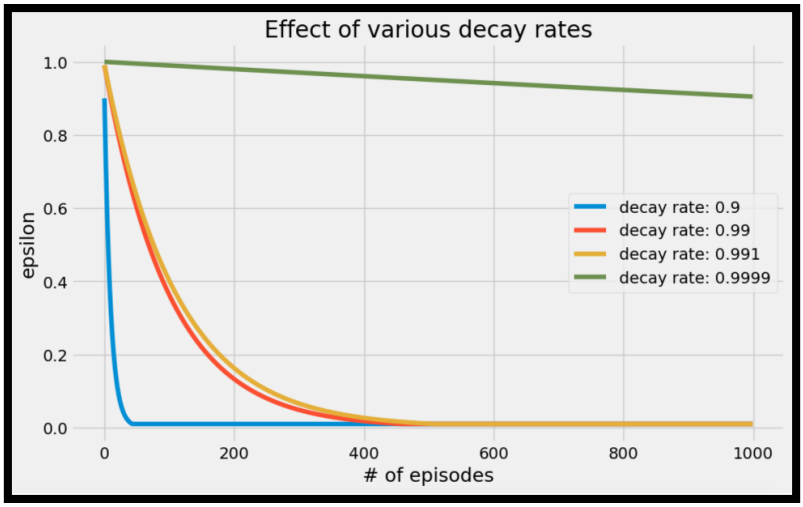


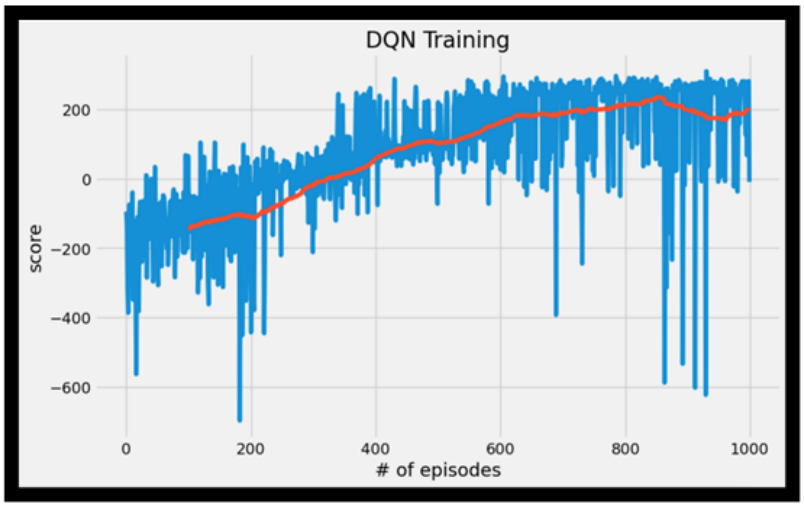
Figure 2: Effect of Decay Rates on Epsilon

The x-axis represents the number of episodes, while the y-axis represents the epsilon value. A faster decay rate leads to a more rapid reduction in epsilon, indicating a quicker transition from exploration to exploitation. As shown, the highest decay rate (0.9) rapidly drops near zero within the first 100 episodes. The red, yellow, and green curves with lower decay rates exhibit more gradual declines in epsilon, with the green curve (decay rate 0.9999) maintaining high levels of exploration even after 1000 episodes.

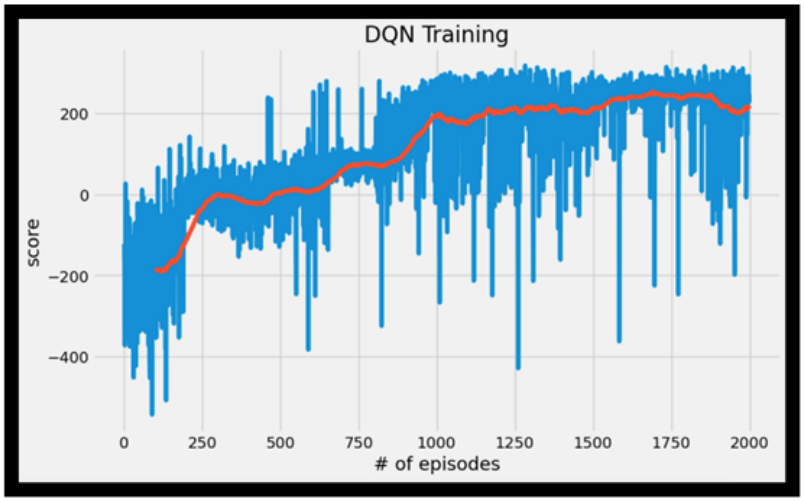
Figure 3 shows the training scores for 1000 (3(a)) and 2000 (3(b)) episodes, with the x-axis representing the number of episodes and the y-axis indicating the score. The blue points represent individual episode scores, while the red line is a rolling average of 100 episodes, smoothing out the fluctuations to highlight the general trend. These graphs illustrate the agent's learning process over time.

Subfigure 3(a) depicts the scores of the agent over 1,000 episodes, showing considerable variability in the agent's performance. Initially, scores fluctuate widely, with several large drops below the zero line, indicating poor performance and exploratory behaviour. As training progresses, the agent's scores improve, reaching a baseline of approximately 200. However, despite this general upward trend, some episodes experience substantial negative scores, with drops below -600. This could be due to factors such as an unstable policy, poor state transitions, or unexpected adverse environmental events.

Subfigure 3(b) illustrates a longer training period being tested to understand how the number of episodes would impact the final performance. The baseline score for an experiment to be considered successful was set to 200. There is a clearer upward trend, with the blue points exhibiting less extreme variability, suggesting that the agent has learned and stabilised. The red line's upward trajectory is more consistent, indicating that the agent has become more adept at the task. However, some significant dips still occur, reflecting potential exploration or sudden



(a)



(b)

Figure 3: DQN Training scores for 1000 (a) and 2000 (b) episodes.

changes in the environment's dynamics. These dips could be attributed to occasional policy shifts as the agent continues to explore or reset its learned behaviours.

Comparing these two graphs, the first notable difference is the training duration: the second graph covers twice as many episodes, allowing for greater learning opportunities and stability. The second graph's reduced fluctuations suggest that extended training helps the agent achieve more consistent performance. The large negative scores in the first graph might result from limited training, causing the agent to explore more aggressively or encounter unexpected situations, leading to performance drops. Moreover, It can be observed that for 1000 episodes, the agent consistently achieves this baseline score after episode 500 with an average score peak around 250, while for 2000 episodes, this is achieved around episode 1300, with the average score of episodes peaking around 300.

The key takeaway is that reinforcement learning balances exploration and exploitation. The second graph shows that longer training with consistent policies can lead to a more stable agent with fewer extreme negative scores. However, occasional negative scores in both graphs indicate the inherent unpredictability of the reinforcement learning process and the need to adapt policies to changing environments continuously.

The experiments of 2000 episodes were recorded and can be viewed here: YouTube: Open AI Gym Lunar Lander environment trained with Deep Q Network

References

[Connolly, 2020] Connolly, J. (2020). After lm nasa lunar lander concepts beyond apollo.

[Dhoble et al., 2020] Dhoble, N., Gu, J., Gunda, I., Kudari, S., Lee, I., Xu, S., and Yang, K. (2020). Lunar lander - deep reinforcement learning, noise robustness, and quantization.

[Gymnasium, 2022] Gymnasium, O. A. (2022). Gymnasium documentation.

[Katnoria, 2017] Katnoria (2017). Deep q network for lunar lander.

Contribution List

- **Olisa Ajufo**: Research and Report for 'Exploration and Exploitation' (Problem 2)
- **Martins Eweniyi**: Research and Report for 'Exploration and Exploitation' (Problem 2)
- **Noel Ogbuagu**: Research and Report for 'Exploration and Exploitation' (Problem 2)
- **Walid Salem**: Code and Report for 'Lunar Lander Agent' Algorithm; (Problem 1)
- **Vasiliki Nikolaidi**: Research and Report for 'Lunar Lander Agent' Algorithm; (Problem 1)