LUNAR LANDER

Saurabh Kumar SC22B146
Sahil Shete SC22B113
Satyajeet Musmade SC22B182
REWARD MAXIMIZERS

Indian Institute of Space Science and Technology
Thiruvananthapuram, Kerala, India 695547

December 16, 2025

Outline

1 Introduction

2 Model and Problem Statement

3 Reinforcement Learning Agent

4 Extensions

5 Performance Analysis

Motivation for the Problem

- **Control Theory Challenge:** Landing a spacecraft is a classic problem requiring precision control under dynamic forces (gravity, thrust).
- **High-Dimensional Control:** Traditional methods become complex as system dynamics increase. Reinforcement Learning (RL) offers a model-free approach.
- **Deep Reinforcement Learning (DRL):** Merges the perception capability of Deep Learning with the decision-making of RL.
- **LunarLander-v3:** A canonical, yet challenging, benchmark environment for testing DRL algorithms like DQN.

The Problem

Train an agent to safely navigate and land a lunar module on a designated landing spot, maximizing cumulative reward while minimizing fuel usage.

Environment Description: LunarLander-v3

The problem is modeled as a Markov Decision Process (MDP) $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$.

- **State Space ($\mathcal{S}$)**: 8 continuous values

$$\text{Box}\begin{pmatrix}[-2.5,\ -2.5,\ -10,\ -10,\ -6.2831855,\ -10,\ 0,\ 0], \\ [2.5,\ 2.5,\ 10,\ 10,\ 6.2831855,\ 10,\ 1,\ 1]\end{pmatrix}$$

  i.e., x position, y position, x velocity, y velocity, angle, angular velocity, left leg contact, right leg contact.

- **Action Space ($\mathcal{A}$)**: 4 discrete actions
  1. Do nothing
  2. Fire main engine
  3. Fire left engine
  4. Fire right engine

Performance Criterion (Goal)

Achieve an average score of ≥ 200.0 over 100 consecutive episodes.

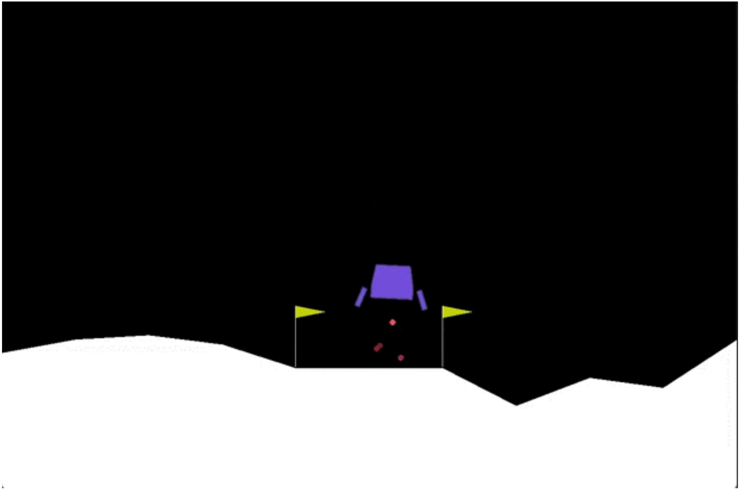

Figure 1: The LunarLander-v3 gym environment

Performance Criterion: Reward Structure

- **Landing/Success:** $+100$ points for coming to rest on the landing pad.
- **Crashing/Failure:** -100 points for contacting the ground outside the pad or non-zero speed.
- **Fuel Usage (Cost):** -0.3 points for firing the main engine, -0.03 for side engines.
- **Leg Contact:** $+10$ points for each leg in contact with the ground.
- **Discount Factor:** 0.99.

The Return

The agent maximizes the discounted cumulative reward (Return):

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

The Deep Q-Network (DQN) Agent

Agent Architecture:

- Uses two identical **Feedforward Neural Networks** (Local and Target)
- Layers:
  $\text{Input}(8) \rightarrow \text{FC}(64, \text{ReLU}) \rightarrow \text{FC}(64, \text{ReLU}) \rightarrow \text{Output}(4)$
- **Experience Replay:** Stores experiences in a capacity $= 10^5$ buffer
- ϵ**-Greedy Policy:** Balances exploration/exploitation

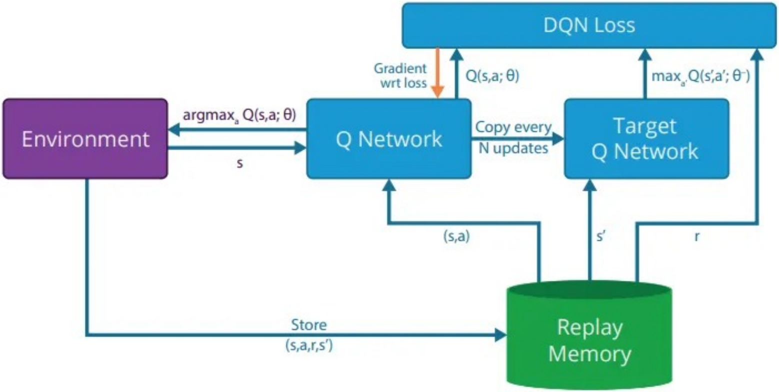

Figure 2: DQN Architecture (Local and Target Networks)

DQN Learning and Updates

- **Q-Target Calculation (Bellman Equation):**

$$Y_t = R_{t+1} + \gamma \max_a Q_{\text{target}}(S_{t+1}, a_{t+1}) \cdot (1 - \text{Done})$$

- **Loss Function:** Mean Squared Error (MSE) between the target and the local network's estimate.

$$\mathcal{L} = \mathbb{E}[(Y_t - Q_{\text{local}}(S_t, A_t))^2]$$

- **Soft Update of Target Network:** Stabilizes training by slowly updating the target weights (θ_{target}).

$$\theta_{\text{target}} \leftarrow \tau\theta_{\text{local}} + (1 - \tau)\theta_{\text{target}}$$

  - τ (interpolation_parameter) $= 10^{-3}$.
  - Updates occur every 4 steps after sampling a minibatch (size $= 100$).

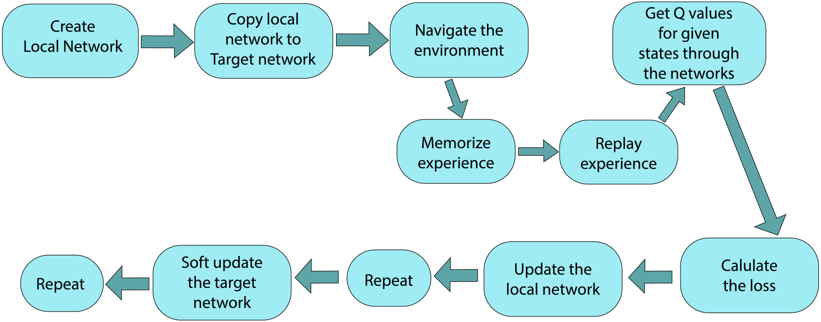

Figure 3: DQN Training process

Key Hyperparameters

- **Learning Rate (Adam Optimizer):** 5×10^{-4}
- **Discount Factor:** 0.99
- **Replay Buffer Size:** 10^5
- ϵ**-Decay Schedule:**
  - Starting ϵ: 1.0
  - Ending ϵ: 0.01
  - Decay Rate: 0.995

Performance Analysis

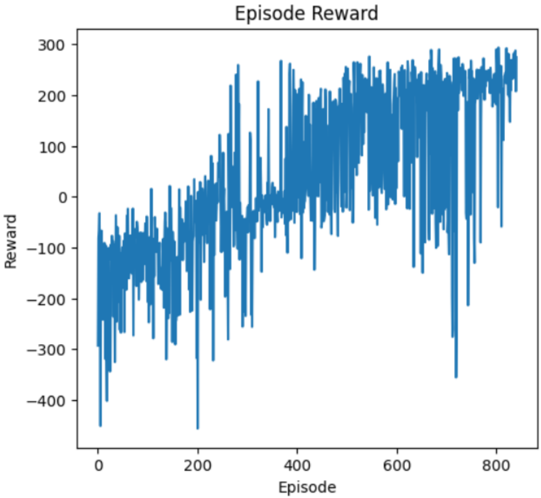

Figure 4: Episode Reward vs. Episode No.

Performance Analysis

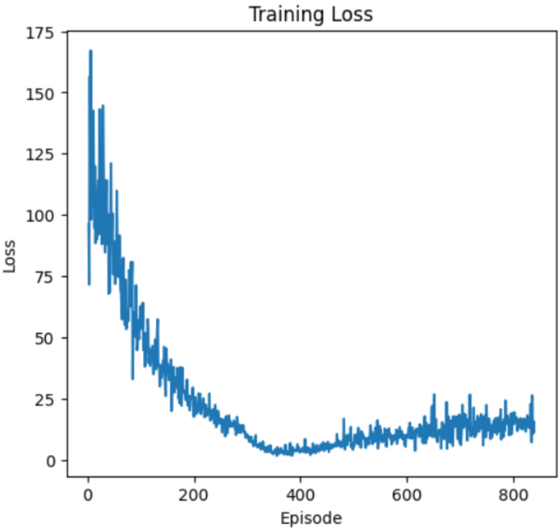

Figure 5: Training Loss vs. Episode No.

Thank you