Project Report

Project: Deep Q-Learning for Lunar Lander

Dhiraj H. Gawhare
Roll Number: 22M0062
M. Tech (Aerodynamics)
IIT Bombay.

Project Title: Deep Q-Learning for Lunar Lander

1. Introduction: The goal of this project is to train an agent to land a lunar lander safely on a landing pad on the surface of the moon using the Deep Q-Learning algorithm with Experience Replay. The project utilizes the OpenAI Gym library to create the Lunar Lander environment and TensorFlow/Keras for building and training the deep learning models.

2. Problem Statement: The Lunar Lander environment presents a challenging reinforcement learning problem where the agent must learn to control a lunar lander to safely land on a landing pad. The agent needs to choose from four discrete actions: do nothing, fire right engine, fire main engine, or fire left engine. The agent's observation space consists of various variables representing the lander's state, such as coordinates, velocities, angles, and leg contacts.

3. Methodology: The project employs the Deep Q-Learning algorithm with Experience Replay to solve the Lunar Lander environment. The key components of the methodology are as follows:

3.1. Q-Network Architecture: A deep neural network, referred to as the Q-Network, is used to approximate the action-value function. The Q-Network is trained to predict the Q-values for each action given the current state as input. The architecture of the Q-Network consists of three Dense layers with relu activation functions.

3.2. Target Network: To stabilize the learning process, a separate target Q-Network is introduced. The target Q-Network has the same architecture as the Q-Network but its weights are updated using a soft update mechanism at regular intervals. This helps mitigate the issue of constantly changing target values during training.

3.3. Experience Replay: Experience Replay involves storing the agent's experiences (state, action, reward, next state, done) in a memory buffer. This buffer is then randomly sampled to create mini-batches for training. Experience Replay reduces correlations between consecutive experiences and allows the agent to learn from a diverse set of experiences.

3.4. Loss Function: The loss function used for training is the Mean-Squared Error (MSE) between the predicted Q-values by the Q-Network and the target Q-values generated by the target Q-Network.

Algorithm 1: Deep Q-Learning with Experience Replay

```
Initialize memory buffer $D$ with capacity $N$
Initialize $Q$-Network with random weights $w$
Initialize target $\hat{Q}$-Network with weights $w^- = w$
for episode $i = 1$ to $M$ do
    Receive initial observation state $S_1$
    for $t = 1$ to $T$ do
        Observe state $S_t$ and choose action $A_t$ using an $\epsilon$-greedy policy
        Take action $A_t$ in the environment, receive reward $R_t$ and next state $S_{t+1}$
        Store experience tuple $(S_t, A_t, R_t, S_{t+1})$ in memory buffer $D$
        Every $C$ steps perform a learning update:
        Sample random mini-batch of experience tuples $(S_j, A_j, R_j, S_{j+1})$ from $D$
        Set $y_j = R_j$ if episode terminates at step $j+1$, otherwise set $y_i = R_j + \gamma \max_{a'} \hat{Q}(s_{j+1}, a')$
        Perform a gradient descent step on $(y_j - Q(s_j, a_j; w))^2$ with respect to the $Q$-Network weights $w$
        Update the weights of the $\hat{Q}$-Network using a soft update
    end
end
```

3.5. Exploration Strategy: An ε-greedy exploration strategy is employed where the agent chooses the action with the highest Q-value with probability (1-ε) and a random action with probability ε. The value of ε starts high (1.0) and gradually decreases during training to encourage exploration early on and exploitation later.

3.6. Training Loop: The agent undergoes training through a series of episodes. In each episode, the agent interacts with the environment, chooses actions, receives rewards, and updates its Q-Network. The total rewards accumulated in each episode are tracked.

4. Training and Results: The agent is trained over a fixed number of episodes (e.g., 2000 episodes) with a maximum number of timesteps per episode (e.g., 1000 timesteps). The training process includes updating the Q-Network, soft-updating the target Q-Network, and adjusting the ε-greedy exploration factor. The project tracks the average total rewards over a moving window of 100 episodes to determine when the environment is solved (average reward ≥ 200).

5. Video Output: The trained agent's performance is visualized by creating a video of the agent's interactions with the environment using the trained Q-Network. The video showcases the agent's ability to land the lunar lander on the landing pad safely.

6. Conclusion: The project demonstrates the successful application of Deep Q-Learning with Experience Replay to solve the challenging Lunar Lander environment. By utilizing a Q-Network to approximate action-values and incorporating techniques like target networks and experience replay, the agent is able to learn an effective policy for landing the lunar lander safely. The project highlights the importance of exploration strategies, neural network architectures, and the stability-enhancing mechanisms in reinforcement learning.

7. Future Improvements:

- Hyperparameter tuning: Experiment with different hyperparameters such as learning rate, discount factor, and mini-batch size to further optimize training.
- Prioritized Experience Replay: Assign different priorities to experiences based on their impact on learning, which can lead to more efficient learning.
- Dueling DQN: Implement Dueling Deep Q-Network architecture to separate the value and advantage streams for better learning.

8. References:

- "Human-level Control Through Deep Reinforcement Learning" by Mnih et al.
- "Continuous Control with Deep Reinforcement Learning" by Lillicrap et al.
- "Playing Atari with Deep Reinforcement Learning" by Mnih et al.
- OpenAI Gym Documentation: https://gym.openai.com/docs/
- TensorFlow Documentation: https://www.tensorflow.org/
- Keras Documentation: https://keras.io/