Lunar Lander OpenAi Gym Final Project

JonPaul Ferzacca
CSPB 3202 Introduction to AI - Professor Truong

Project Overview

This may have been the most difficult project yet. The project undertakes the challenge of applying Deep Reinforcement Learning, specifically through the implementation of a Deep Q-Network (DQN), in the 'LunarLander-v2' environment. The agent aims to learn the optimal actions required to land a spacecraft on the lunar surface. It simulates a complex control task involving navigation and soft landing, requiring the agent to manage both linear and angular movements while considering environmental physics. It took a ton of different styles and approaches to maximize the potential.

Approach

Environment Description:
'LunarLander-v2' is a dynamic and rich environment where a lander must be guided to land between two flags. The simulation provides continuous state feedback, including the lander's position, velocity, and angle.

Model Architecture:
The DQN model is built using PyTorch, featuring fully connected layers with ReLU activation. The network architecture includes two hidden layers, each with 64 neurons, designed to process the state information and estimate action values.

Training Mechanics:
The agent uses an epsilon-greedy strategy for exploration, where the value of epsilon decays over time, allowing a transition from random exploration to exploiting learned values.
Experiences (state, action, reward, next state, done) are stored in a memory buffer, enabling learning from past experiences and stabilizing the training process.
The agent periodically updates its policy network based on mini-batches sampled from the experience replay buffer.

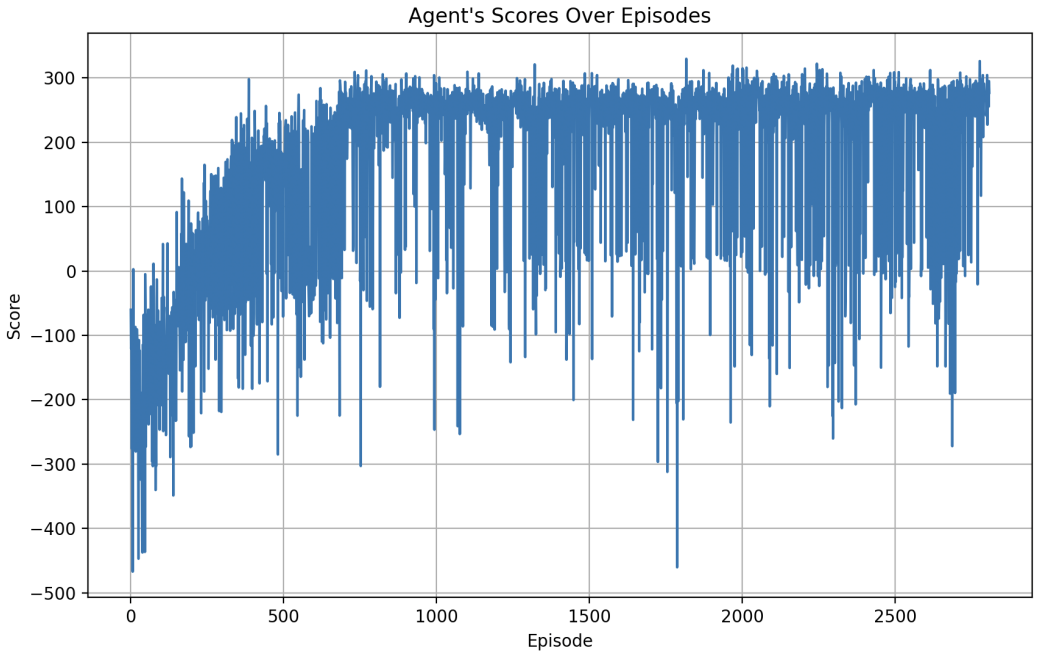
Loss Function and Optimization:
The Mean Squared Error (MSE) loss is utilized, and the Adam optimizer is employed for network training.

Results and Analysis

Training Dynamics:
The agent's learning progression was monitored over 5000 episodes, with the target of achieving an average score of 250 over consecutive episodes.
Performance metrics include the score per episode and the average score over the last 100 episodes.
Epsilon decayed from 1.0 to 0.1, facilitating the agent's shift from exploration to exploitation.

```
:08,  5.96episode/s, ScorTraining Progress:  56%|▌| 2801/5000 [09:19<06:08,  5.96episode/
s, ScorTraining Progress:  56%|▌| 2801/5000 [09:19<06:08,  5.96episode/s, ScorTraining Pr
ogress:  56%|▌| 2803/5000 [09:19<05:42,  6.41episode/s, ScorTraining Progress:  56%|▌| 28
03/5000 [09:19<05:42,  6.41episode/s, ScorTraining Progress:  56%|▌| 2803/5000 [09:19<05:
42,  6.41episode/s, ScorTraining Progress:  56%|▌| 2805/5000 [09:19<04:46,  7.65episode/s
, ScorTraining Progress:  56%|▌| 2805/5000 [09:19<04:46,  7.65episode/s, ScorTraining Pro
gress:  56%|▌| 2805/5000 [09:19<04:46,  7.65episode/s, ScorTraining Progress:  56%|▌| 280
7/5000 [09:19<04:17,  8.51episode/s, ScorTraining Progress:  56%|▌| 2807/5000 [09:20<04:1
7,  8.51episode/s, ScorTraining Progress:  56%|▌| 2808/5000 [09:20<05:20,  6.83episode/s,
 ScorTraining Progress:  56%|▌| 2808/5000 [09:20<05:20,  6.83episode/s, Score: 276.92, A
Target score achieved!
Training Progress:  56%|▌| 2808/5000 [09:20<07:17,  5.01episode/s, Score: 276.92, A
2023-12-18 22:36:05.885 Python[83458:8192021] +[CATransaction synchronize] called within
transaction
Test run score: 285.9319834554128
Test run score: 172.9090288384415
Test run score: 268.6970202414477
Test run score: 281.6233830761332
Test run score: 280.7195702924028
Test run score: 256.07292838887076
Test run score: 244.9743323717179
Test run score: 255.41620066876078
Test run score: 178.71636916394192
Test run score: 265.1309702444498
```

Agent's Learning Curve:
The scores over episodes depict the agent's improvement and stabilization in performance.
Initial fluctuations in scores are expected due to exploration.
A plot of these scores provides visual insight into the learning efficiency and the consistency of the agent's performance over time.



Learning Progression

At 56% completion of the training (2808 out of 5000 episodes), the agent achieved a notable score of 276.92. This indicates a significant learning progression, as the agent is already surpassing the desired average score target of 250.

Training Efficiency

The steady increase in score, reaching 276.92 in just over half the total planned episodes, suggests effective learning strategies employed by the DQN agent. The epsilon decay strategy, allowing a gradual shift from exploration to exploitation, seems to be working efficiently.

Consistency in Performance

The continuous improvement in scores implies that the agent is not only learning but also becoming consistent in applying its learned strategies to complete the landing task.

Conclusion and Future Work

Project Insights:
This project showcases the capabilities of DQN in mastering a complex control task within a simulated environment. The agent's success in learning to land on the lunar surface underscores the potential of deep reinforcement learning in solving real-world analogous problems.

Recommendations for Future Enhancements:
Implementing advanced variants of DQN, such as Double DQN or Dueling DQN, could address issues like overestimation of Q-values.
Experimentation with different network architectures or hyperparameters could further optimize performance.
Applying the model to other challenging environments or introducing variable environmental conditions in 'LunarLander-v2' could test the adaptability and robustness of the agent.

Source Code and Documentation:
https://github.com/JPFerzacca/CSPB3202_Final_Project.git

Visual Demonstrations: A collection of video clips to see the model perform are uploaded to Git Hub.

Resources:

https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html
https://github.com/yuchen071/DQN-for-LunarLander-v2
https://www.youtube.com/watch?v=p0rGjAgykOU
https://gymnasium.farama.org/
https://github.com/Farama-Foundation/Gymnasium
https://github.com/openai/retro
https://pypi.org/project/torch/