Deep Q-Learning for Space Invaders

Finn Rehnert
University of Applied Sciences Ulm
Advanced Machine Learning Project
refinn01@thu.de

***Abstract*—todo write abstract**

I. Introduction

The Hook: Briefly define Reinforcement Learning (an agent learning by interacting with an environment) and the specific challenge of high-dimensional inputs (pixels).

Motivation: Explicitly state: "Since RL was not covered in the course lectures, this project serves as an explorative study into Deep Reinforcement Learning."

The Goal: To replicate the success of the 2013 DeepMind paper on the specific environment of Space Invaders. [1]

A. Scenario

Describe the Space Invaders environment (part of the OpenAI Gym/Ale).

Define the State Space: Raw pixels (RGB images).

Define the Action Space: Discrete actions (Move Left, Move Right, Shoot, No-op).

Define the Reward Function: Points scored for killing aliens.

B. Structure of the Paper

II. Literature Review

Note: This is where you explain the "Why" behind the math.

A. From Tabular Learning to Function Approximation

Explain traditional Q-Learning (using a Q-table).

Explain the limitation: The "Curse of Dimensionality." You cannot have a table row for every possible pixel combination in Space Invaders.

Introduce the solution: Using a function approximator (Neural Network) to estimate Q-values.

B. The Deep Q-Network (DQN) Breakthrough

Discuss the Mnih et al. (2013) paper.

Highlight the two key innovations that stabilized training (which you implemented):

Experience Replay (breaking correlation between consecutive frames).

Target Networks (stabilizing the moving target).

III. Concept for problem solving

Define the Bellman Equation.

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

Explain the Loss Function. You are minimizing the Mean Squared Error between the predicted Q-value and the target Q-value. Explain ϵ-greedy exploration: How the agent balances exploring random moves vs. exploiting known best moves.

A. Overall Solution Strategy

Mention the tools used: Python, PyTorch/TensorFlow, OpenAI Gym. High-level data flow: Environment $\rightarrow$ Wrapper $\rightarrow$ Buffer $\rightarrow$ Network $\rightarrow$ Optimizer.

B. Preprocessing and Wrappers

Crucial for Atari: Explain how you processed the raw inputs. Grayscale (3 channels $\rightarrow$ 1 channel). Resizing (e.g., to 84x84).

Frame Stacking: Explain why this is needed (a single image doesn't show direction/velocity; stacking 4 frames gives temporal context).

C. Network Architecture and Training Loop

Describe the CNN architecture (Convolutional layers $\rightarrow$ Fully connected layers $\rightarrow$ Output nodes for each action). Describe the training loop (Sample batch $\rightarrow$ Calculate Loss $\rightarrow$ Backprop). Mention Hyperparameters (Learning rate, Gamma, Buffer size).

IV. Results and Evaluation

Critique: Your original skeleton skipped this, but for an ML project, this is mandatory. You must insert this section before ”Further Steps.”

Training Curve: Plot Episode (x-axis) vs. Average Reward (y-axis).

Qualitative Analysis: Does the agent actually play well? Does it learn to hide behind the shields?

Comparison: Compare the trained agent against a ”Random Agent” (one that just presses buttons randomly).

V. Conclusion

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing atari with deep reinforcement learning,” 2013. [Online]. Available: https://arxiv.org/abs/1312.5602