Project 4 — Reinforcement Learning in Gym/Gymnasium: Pong with Deep Q-Learning

Overview

This project trains and evaluates a Deep Q-Learning (DQN) agent in a custom Pong environment (`SimplePongEnv`) rendered using Pillow with a 6-dimensional numeric state. The goal is to implement a lightweight approximate Deep RL solution, analyze reward/score learning curves, and generate stable 15-second continuous evaluation demos without flashing or corrupted frames.

Environment Description

• Environment: Custom `SimplePongEnv` built using Pillow rendering (static black background, no flashing lights)
• Observation Space (6 normalized floats): ball x/y position, x/y velocity, left/right paddle y-positions
• Action Space: `Discrete(3)` → 0=stay, 1=up, 2=down (agent controls the left paddle)
• Opponent: Rule-based right paddle tracking (for stable evaluation)
• Reward: +1 for paddle hit, –1 for miss or ball exit (episode termination on miss/exit)

Model Selection

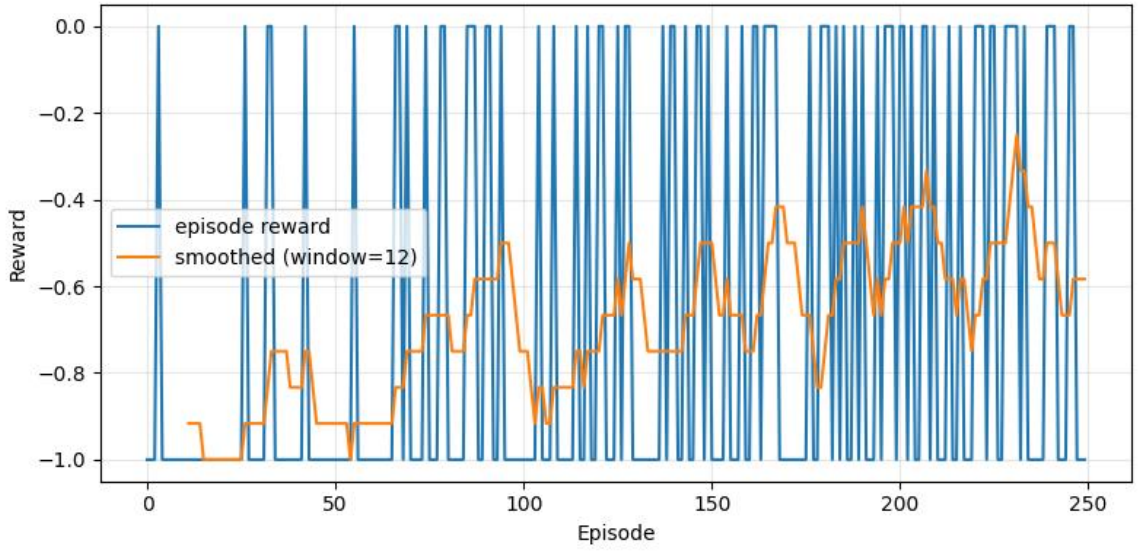
• Algorithm: Deep Q-Network (DQN)
• Network: Multi-Layer Perceptron (MLP) → 6 → 128 → 128 → 3 Q-values
• Activations: ReLU
• Loss: MSE toward Bellman Q-target
• Optimizer: Adam (lr = 1×10^{-3})
• Discount factor: γ = 0.99
• Replay Buffer: 10,000 transitions
• Exploration: ε-greedy during training (1.0 → 0.05 decay; deterministic at evaluation)
• Target-network sync every 2 episodes
• Final model weights saved (`.pth` file), CPU fallback supported

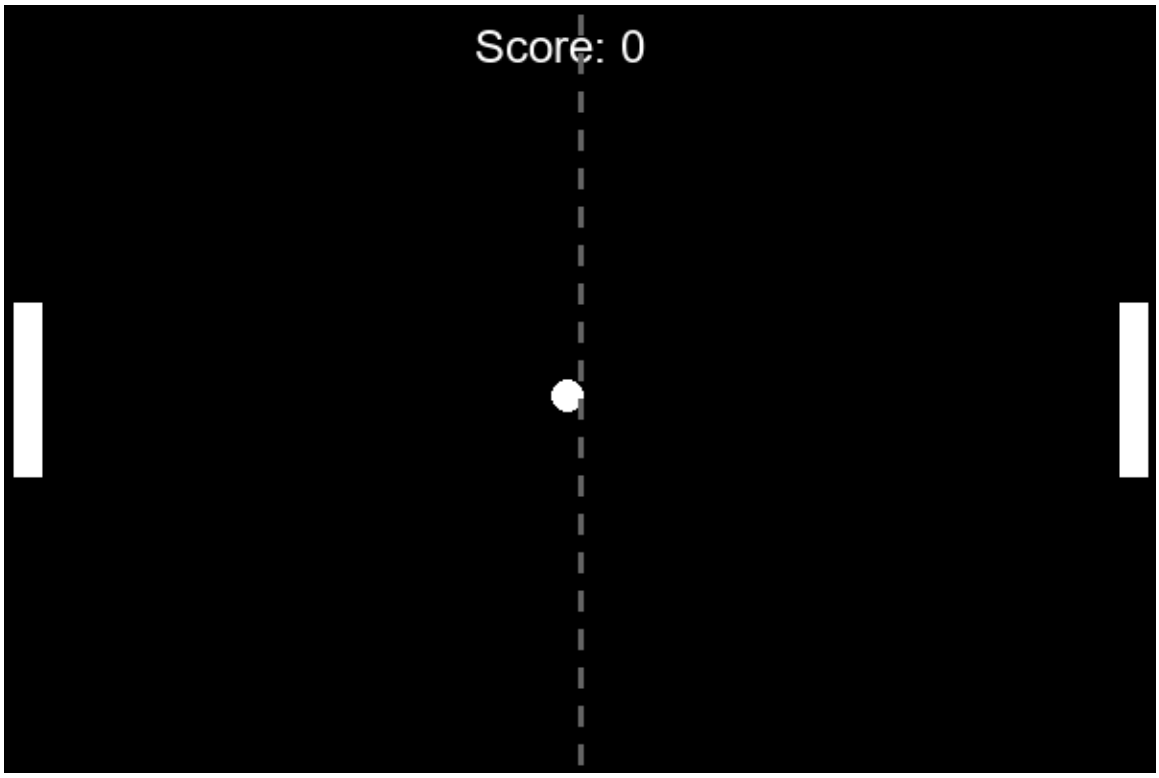
Approach & Troubleshooting

Training approach:
1. Initialize Gymnasium-compatible environment
2. Select actions using ε-decay exploration
3. Store transitions (s, a, r, s', done) into replay memory
4. Train mini-batch Q-updates each step using Bellman targets
5. Periodically update frozen target network
6. Save trained model weights after training

Major issues resolved:
• Atari wrapper import errors (`FrameStack`, `RecordVideo`) → replaced with custom Pillow frame pipeline
• Torch install include/header failure on Windows → rebuilt `.venv` using CPU-safe wheel
• GIF corruption/glitching → Pillow export uses `optimize=False`, `disposal=2`, padded 15s frames
• `AttributeError: ball missing on eval` → guaranteed initialization in `reset()`
• Flashing lights → static black backgrounds, no alternating colors, no delta flicker

Results & Interpretation





Iterative Improvements Summary

• Target network updates added for stable Q-learning
• Ball velocity multipliers clipped to prevent paddle tunneling
• Frame padding used for fixed 15s demo (no mid-episode resets)
• CPU-safe fallback integrated for torch-free execution

Conclusion & Reflection

A compact MLP-based DQN can learn to control the Pong paddle using normalized numeric state inputs. Major roadblocks included missing Atari wrappers and torch installation failures on Windows, which were solved by simplifying dependencies and using CPU-stable libraries. Future extensions may include CNN-based vision observations, PPO policy gradient control, or self-play opponents.

Future Improvements / Ideas

• Switch to CNN for vision-based state
• Upgrade to Double or Dueling DQN architectures
• Add self-play or curriculum learning
• Test on additional environments for generalization

References

https://www.python.org/

https://pypi.org/project/pillow/

https://gymnasium.farama.org/index.html

https://numpy.org/

https://github.com/imageio/imageio

https://pytorch.org/

https://docs.python.org/3/library/venv.html

https://www.geeksforgeeks.org/deep-learning/deep-q-learning/

https://docs.pytorch.org/docs/stable/generated/torch.nn.ReLU.html

https://apxml.com/courses/intermediate-reinforcement-learning/chapter-2-deep-q-networks-dqn/dqn-loss-function