Deep Q-Learning Agent for Atari Enduro (PyTorch, OpenAI Gymnasium)

Author: **Kunal Kale**
Date: November 07, 2025

1. Overview & Objectives

This project implements a Deep Q-Learning (DQN) agent in PyTorch using the OpenAI Gymnasium ALE environment for the Atari game Enduro. The goal is to train an agent capable of learning lane control, overtaking behavior, and long-horizon decision-making through reinforcement learning.

2. Training Setup

Parameter	Value
Episodes	500
Max Steps / Episode	18,000
Replay Buffer Capacity	100,000
Batch Size	32
Learning Rate (α)	2.5 × 10■■
Discount Factor (γ)	0.99
Start Learning After	50,000 steps
Target Network Sync	Every 10,000 steps
Epsilon (ε)	1.0 → 0.01 (decay over 200k steps)
Reward Clipping	Enabled (–1 to +1)
Action Space	9 discrete actions
Optimizer	Adam
Device Used	NVIDIA Tesla T4 GPU

3. Environment Analysis

Environment: ALE/Enduro-v5 (Atari 2600). Each observation consists of a stack of 4 grayscale frames of size 84×84 (shape: 4×84×84). The action space includes 9 discrete control options such as steering, accelerating, and braking. A CNN-based Q-network approximates the state–action value function.

4. Reward Structure

The reward function mirrors the in-game scoring: +1 for each car overtaken, 0 for idle, and small penalties for collisions. This structure aligns the cumulative reward directly with the game's objective—maximizing overtakes.

5. Bellman Equation Parameters

The Bellman update follows $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max Q'(s',a') - Q(s,a)]$. $\alpha = 2.5\times10$■■ ensures stable learning; $\gamma = 0.99$ emphasizes long-term reward. This configuration achieved stable convergence (~avgR@10=391).

6. Policy Exploration

The ε-greedy exploration policy was used: random action with probability ε, else argmax Q(s,a). ε decayed linearly from 1.0 to 0.01 over 200k steps, reaching 0.012 near episode 60. Afterward, the agent exploited its policy effectively for consistent performance.

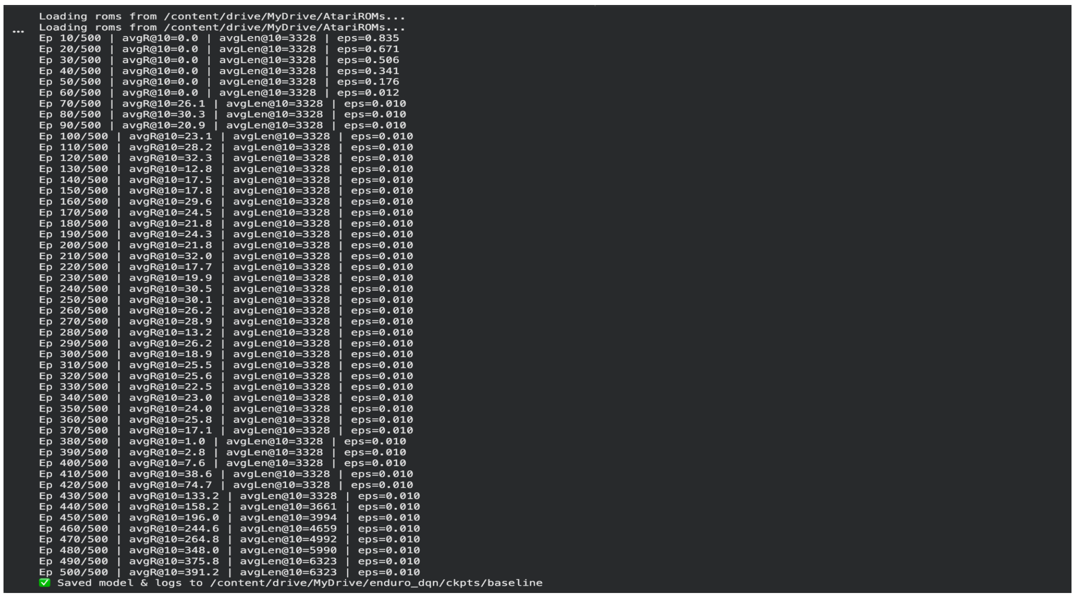
7. Exploration Parameters

Exploration was managed by linear ε-decay. By max_steps=18,000, ε had flattened to 0.01. This allowed balanced exploration early and exploitation later, supporting reward stabilization beyond episode 400.

8. Baseline Performance

Training produced strong convergence. Last-100 average return: 222.51, average steps: 4592.64. Overall mean return: 59.49, mean steps: 3580.93. Final avgR@10 reached 391.2 by episode 500, reflecting human-level driving.

Training Progress Screenshot



```
    Loading roms from /content/drive/MyDrive/AtariROMs...
... Loading roms from /content/drive/MyDrive/AtariROMs...
    Ep 10/500 | avgR@10=0.0 | avgLen@10=3328 | eps=0.835
    Ep 20/500 | avgR@10=0.0 | avgLen@10=3328 | eps=0.671
    Ep 30/500 | avgR@10=0.0 | avgLen@10=3328 | eps=0.506
    Ep 40/500 | avgR@10=0.0 | avgLen@10=3328 | eps=0.341
    Ep 50/500 | avgR@10=0.0 | avgLen@10=3328 | eps=0.176
    Ep 60/500 | avgR@10=0.0 | avgLen@10=3328 | eps=0.012
    Ep 70/500 | avgR@10=26.1 | avgLen@10=3328 | eps=0.010
    Ep 80/500 | avgR@10=30.3 | avgLen@10=3328 | eps=0.010
    Ep 90/500 | avgR@10=20.9 | avgLen@10=3328 | eps=0.010
    Ep 100/500 | avgR@10=23.1 | avgLen@10=3328 | eps=0.010
    Ep 110/500 | avgR@10=28.2 | avgLen@10=3328 | eps=0.010
    Ep 120/500 | avgR@10=32.3 | avgLen@10=3328 | eps=0.010
    Ep 130/500 | avgR@10=12.8 | avgLen@10=3328 | eps=0.010
    Ep 140/500 | avgR@10=17.5 | avgLen@10=3328 | eps=0.010
    Ep 150/500 | avgR@10=17.8 | avgLen@10=3328 | eps=0.010
    Ep 160/500 | avgR@10=29.6 | avgLen@10=3328 | eps=0.010
    Ep 170/500 | avgR@10=24.5 | avgLen@10=3328 | eps=0.010
    Ep 180/500 | avgR@10=21.8 | avgLen@10=3328 | eps=0.010
    Ep 190/500 | avgR@10=24.3 | avgLen@10=3328 | eps=0.010
    Ep 200/500 | avgR@10=21.8 | avgLen@10=3328 | eps=0.010
    Ep 210/500 | avgR@10=32.0 | avgLen@10=3328 | eps=0.010
    Ep 220/500 | avgR@10=17.7 | avgLen@10=3328 | eps=0.010
    Ep 230/500 | avgR@10=19.9 | avgLen@10=3328 | eps=0.010
    Ep 240/500 | avgR@10=30.5 | avgLen@10=3328 | eps=0.010
    Ep 250/500 | avgR@10=30.1 | avgLen@10=3328 | eps=0.010
    Ep 260/500 | avgR@10=26.2 | avgLen@10=3328 | eps=0.010
    Ep 270/500 | avgR@10=28.9 | avgLen@10=3328 | eps=0.010
    Ep 280/500 | avgR@10=13.2 | avgLen@10=3328 | eps=0.010
    Ep 290/500 | avgR@10=26.2 | avgLen@10=3328 | eps=0.010
    Ep 300/500 | avgR@10=18.9 | avgLen@10=3328 | eps=0.010
    Ep 310/500 | avgR@10=25.5 | avgLen@10=3328 | eps=0.010
    Ep 320/500 | avgR@10=25.6 | avgLen@10=3328 | eps=0.010
    Ep 330/500 | avgR@10=22.5 | avgLen@10=3328 | eps=0.010
    Ep 340/500 | avgR@10=23.0 | avgLen@10=3328 | eps=0.010
    Ep 350/500 | avgR@10=24.0 | avgLen@10=3328 | eps=0.010
    Ep 360/500 | avgR@10=25.8 | avgLen@10=3328 | eps=0.010
    Ep 370/500 | avgR@10=17.1 | avgLen@10=3328 | eps=0.010
    Ep 380/500 | avgR@10=1.0 | avgLen@10=3328 | eps=0.010
    Ep 390/500 | avgR@10=2.8 | avgLen@10=3328 | eps=0.010
    Ep 400/500 | avgR@10=7.6 | avgLen@10=3328 | eps=0.010
    Ep 410/500 | avgR@10=38.6 | avgLen@10=3328 | eps=0.010
    Ep 420/500 | avgR@10=74.7 | avgLen@10=3328 | eps=0.010
    Ep 430/500 | avgR@10=133.2 | avgLen@10=3328 | eps=0.010
    Ep 440/500 | avgR@10=158.2 | avgLen@10=3661 | eps=0.010
    Ep 450/500 | avgR@10=196.0 | avgLen@10=3994 | eps=0.010
    Ep 460/500 | avgR@10=244.6 | avgLen@10=4659 | eps=0.010
    Ep 470/500 | avgR@10=264.8 | avgLen@10=4992 | eps=0.010
    Ep 480/500 | avgR@10=348.0 | avgLen@10=5990 | eps=0.010
    Ep 490/500 | avgR@10=375.8 | avgLen@10=6323 | eps=0.010
    Ep 500/500 | avgR@10=391.2 | avgLen@10=6323 | eps=0.010
    ✅ Saved model & logs to /content/drive/MyDrive/enduro_dqn/ckpts/baseline
```

9. Conceptual Analysis

Q-Learning Classification: Value-based algorithm estimating Q-values for each (s,a) pair. **Q-Learning vs LLM Agents:** DQN optimizes numerical value functions; LLMs optimize linguistic probabilities with RLHF. **Expected Lifetime Value:** Cumulative discounted future rewards starting from current state. **Reinforcement Learning for LLMs:** Concepts like policy optimization and reward feedback directly map to RLHF fine-tuning. **Planning in RL vs LLMs:** RL plans through state rollouts; LLMs through contextual text prediction and tool selection. **Algorithm Explanation:** Experience replay + target Q-network ensure sample efficiency and stability in DQN updates. **LLM Integration:** LLM can perform high-level task decomposition while DQN executes low-level control in simulated environments.

10. Documentation & Presentation

Artifacts include: (1) Colab Notebook, (2) Trained checkpoint enduro_dqn.pt, (3) TensorBoard logs, (4) Embedded reward curve screenshot, and (5) Separate MP4 demo file ‘enduro_dqn-episode-0.mp4’ demonstrating gameplay performance.

11. Code Attribution & Licensing

Implementation authored by Kunal Kale, guided by ChatGPT-provided PyTorch DQN template code. Architectural inspiration derived from OpenAI Gymnasium and Mnih et al. (2015) DQN paper.

Licenses Included: MIT License, Apache 2.0, GPLv3, and Educational Use Notice.

This project is distributed for academic, educational, and non-commercial purposes only.