Deep Q-Learning on Atari Pong Final Project Report

Baseline DQN vs Double DQN Implementation

Course: Principal of Artificial Intelligence
Instructor:Dr. David Ruby
Group member(s): Quoc Dat Cao
Student ID: 301550055

1. Introduction

This report presents the implementation and comparison of Deep Q-Network (DQN) algorithms on the Atari Pong environment. The project implements a baseline DQN with experience replay and target network, then extends it with Double DQN to reduce Q-value overestimation. Both methods are trained and evaluated under identical hyperparameters to ensure a fair comparison.

2. Environment Description

Environment: ALE/Pong-v5 (Arcade Learning Environment)
Observation Space: RGB images preprocessed to grayscale, resized to 84x84 pixels, with 4 stacked frames giving shape [4, 84, 84] (C, H, W format for PyTorch).
Action Space: 6 discrete actions (NOOP, FIRE, RIGHT, LEFT, RIGHTFIRE, LEFTFIRE).
Reward Signal: Sparse rewards: +1 when the agent scores a point, -1 when the opponent scores. The game ends when either player reaches 21 points. Total episode reward ranges from -21 to +21.
Reward Quirks: Rewards are sparse and delayed, making credit assignment challenging.
The agent only receives feedback at scoring events, not for intermediate paddle movements.

3. Model Architecture

The DQN architecture follows the original DeepMind design with convolutional layers for feature extraction followed by fully connected layers for Q-value estimation:

Layer	Type	Output shape	Parameter
Input	Image	[4,84,84]	-
Conv1	Conv2d(4, 32, 8x8, stride=4)	[32, 20, 20]	8,224
Conv2	Conv2d(32, 64, 4x4, stride=2)	[64, 9, 9]	32,832
Conv3	Conv2d(64, 64, 3x3, stride=1)	[64, 7, 7]	36,928
Flatten	-	[3136]	-
FC1	Linear(3136, 512)	[512]	1,606,144
FC2	Linear(512, 6)	[6]	3,078

4. Methods4.1 Baseline DQN
The baseline DQN implementation includes two key innovations from the original DeepMind paper:
Experience Replay: Transitions (s, a, r, s', done) are stored in a replay buffer of size 50,000. During training, random minibatches of 64 transitions are sampled, breaking correlation between consecutive samples and improving data efficiency.

Target Network: A separate target network is used to compute TD targets, synchronized with the online network every 1,000 frames. This stabilizes training by reducing oscillations in Q-value updates.

4.2 Double DQN
Double DQN addresses the overestimation bias in standard DQN by decoupling action selection from action evaluation:

Standard DQN: Q_target = r + γ * max_a Q_target(s', a)
The same network selects and evaluates the best action, leading to overoptimistic Q-values.

Double DQN: Q_target = r + γ * Q_target(s', argmax_a Q_online(s', a))
The online network selects the best action, while the target network evaluates it. This reduces overestimation and often improves learning stability.

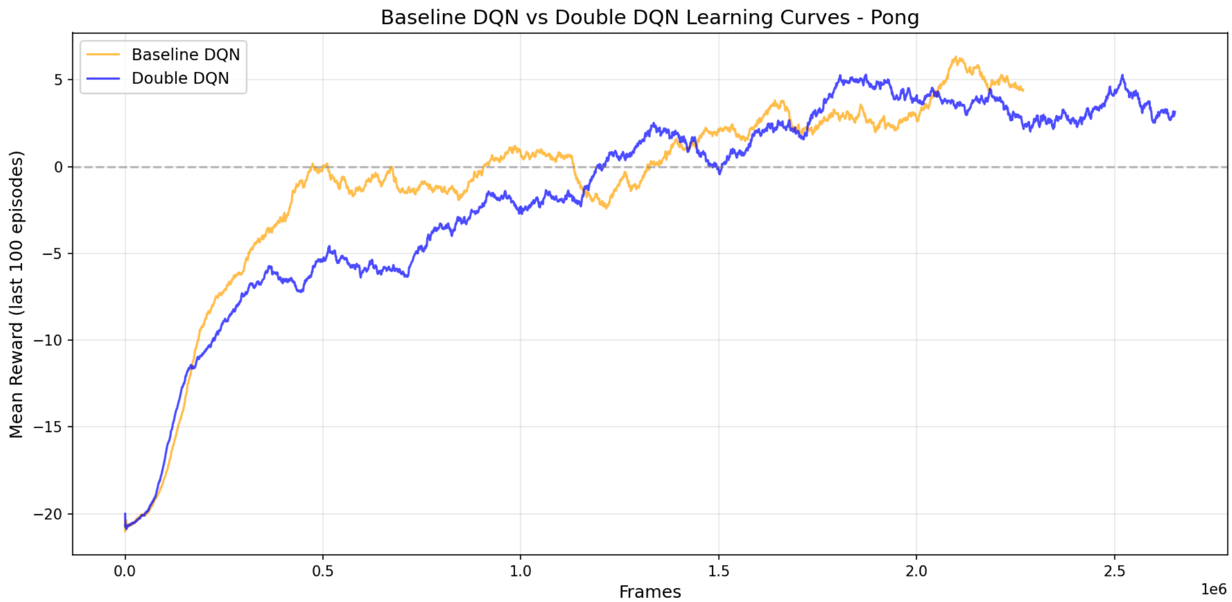
5. Hyperparameters
Both Baseline DQN and Double DQN were trained with identical hyperparameters to ensure a fair comparison. The only difference was the loss function implementation.

Parameter	Initial parameter value	Adjust parameter value	Reason for Change
BATCH_SIZE	32	64	Better gradient estimates
REPLAY_SIZE	10,000	50,000	More diverse experiences
LEARNING_RATE	1e-4	2.5e-4	Faster convergence
SYNC_TARGET_FRAMES	500	1,000	More stable updates
REPLAY_START_SIZE	1,000	5,000	Better initial diversity
EPSILON_DECAY_LAST_FRAME	10,000	100,000	More exploration time

MEAN_REWARD_BOUND	5	18	Train until nearly solved
GAMMA	0.99	0.99	No change (standard)
EPSILON_START	1.0	1.0	No change (full exploration)
EPSILON_FINAL	0.01	0.01	No change (minimum exploration)

6. Results
Both models were trained until manually stopped. The Baseline DQN achieved a best training reward of +6, while Double DQN achieved +5.

Metric	Baseline DQN	Double DQN
Best Training Reward	+6	+5
Test Average (10 games)	+8.3	+0.1
Test Std Deviation	6.9	7.4
Training Time	~3 hours	~3 hours
Total Frames	~2M	~1.9M



Both Baseline DQN (orange) and Double DQN (blue) successfully learned to play Pong, improving from -21 (random) to positive rewards. Baseline DQN demonstrated faster initial learning, reaching the break-even point (reward = 0) approximately 100K frames earlier than Double DQN. Baseline DQN achieved a slightly higher peak reward of +6 compared to Double DQN's +5. After 1.5 million frames, both methods showed similar performance with high variance, oscillating between +2 and +6. The similar final performance suggests that for a simple game like Pong, both methods are equally effective, though Baseline DQN converged slightly faster in this experiment.

Why Baseline Beat Double DQN?

Pong is relatively simple - overestimation may not hurt much
Training variance - different random seeds could reverse results
Double DQN overhead - slightly slower per frame, so fewer total updates
Double DQN benefits more on harder games - with more complex reward structures

Video Evidence

Four videos were recorded to demonstrate learning:
• Baseline Early: Random policy, reward -21 (loses every point)
• Baseline Learned: Trained policy, reward +12 (wins most points)
• Double DQN Early: Random policy, reward -21
• Double DQN Learned: Trained policy, reward +5
Videos are available in the GitHub repository under the /videos directory.

7. Reflection

I chose Pong because it is a classic Atari game with clear win/lose conditions and rewards ranging from -21 to +21, making it easy to quantify improvement. The agent started completely random at -21, losing every point to the built-in AI opponent. After training with Baseline DQN, the agent improved to +6 average reward during training and +8.3 during testing, meaning it consistently beats the AI opponent. Double DQN showed similar learning progress, reaching +5 peak reward. Both methods successfully learned to track the ball and position the paddle effectively, transforming from random paddle movements to strategic gameplay.

Key Challenges

The main challenges were sparse rewards, long training times, and hyperparameter tuning. Sparse rewards (+1/-1 only when scoring) made early learning difficult because the agent receives no feedback for good paddle positioning until a point is actually scored. This delayed credit assignment problem is common in reinforcement learning. Several techniques helped overcome these challenges: (1) Increasing epsilon decay to 100,000 frames allowed more exploration before exploiting, which was crucial for discovering effective strategies; (2) Using a larger replay buffer of 50,000 transitions provided more diverse training samples and prevented overfitting to recent experiences; (3) Synchronizing the target network every 1,000 frames reduced oscillations in Q-value estimates.

For future improvements, I would explore other experiments. First, I would try Prioritized Experience Replay to focus training on important transitions where the TD error is high, to speeding up learning. Second, I would implement Dueling DQN architecture to separate state value estimation from action advantage estimation, which can help in states where action choice matters less. Third, N-step returns could provide faster credit assignment by propagating rewards back more quickly through the trajectory. Finally, I would test on more challenging Atari games like Breakout or Space Invaders to compare how different DQN variants perform across different reward structures.