CSCI 166 Final Project Report

DQN and Double DQN on Atari Pong

Isac Lopez
Fall 2025

Deep Reinforcement Learning (DRL) has demonstrated impressive capabilities on challenging sequential decision-making problems, especially in Atari games where an agent must learn good policies directly from raw pixel observations. One of the foundational DRL algorithms is the Deep Q-Network (DQN), which combines Q-learning with deep convolutional neural networks and experience replay. DQN was the first method to achieve human-level performance on a substantial number of Atari games, making it an excellent foundation for experimentation.

In this project, I implemented a baseline DQN and a Double DQN variant to evaluate how architectural and algorithmic improvements influence learning stability and performance. Pong was selected as the test environment because it is widely used for benchmarking RL algorithms, has a simple discrete action space, and provides a clear visual representation of learning progress. The project includes analysis of learning curves, training behavior, early vs. learned gameplay videos, and a written reflection on challenges and potential improvements.

Environment Description

For all experiments, I used the Atari 2600 Pong environment from Gymnasium:
ALE/Pong-v5

Observation Space:

The environment provides grayscale pixel frames which, after preprocessing, become:

- **Shape:** (4, 84, 84)
- **Type:** uint8
- **Content:** last 4 stacked frames to capture motion
- Processed using standard Atari wrappers (down-sampling, grayscaling, frame stacking).

This setup allows the convolutional neural network to extract spatial and temporal features from recent game history.

Action Space:

The action space is discrete and small:

- 0: NOOP
- 1: FIRE / START
- 2: UP
- 3: DOWN

Depending on wrappers, FIRE may be skipped; the meaningful actions for Pong are UP, DOWN, and sometimes NOOP.

Reward Structure:

Pong has a sparse, delayed reward signal:

- +1 for scoring a point
- –1 when the opponent scores
- 0 otherwise

Scores occur infrequently, so the agent must learn from long sequences of zero-reward transitions. Episodes end when either side reaches the win condition (typically +21 or –21).

Challenges of the Environment:

- Sparse rewards delay learning.
- The agent can play many frames before receiving any reward signal.
- Credit assignment is difficult: useful behaviors (tracking ball motion) must emerge before rewards become meaningful.

Methods

Baseline DQN Implementation:

The baseline DQN follows the original architecture used in the 2015 Nature paper:

Network Architecture

- **Input:** (4, 84, 84) stacked frames
- **Layers:**
    - Conv2d: 32 filters, 8×8 kernel, stride 4
    - Conv2d: 64 filters, 4×4 kernel, stride 2
    - Conv2d: 64 filters, 3×3 kernel, stride 1
    - Flatten
    - Fully connected (512 units, ReLU)
    - Output layer: |A| Q-values for each action

Key Components

- **Experience Replay Buffer:**
    - Size = 10,000 (memory-friendly for Colab)
    - Stores transitions: (state, action, reward, done, next_state)
- **Target Network:**
    - Updated every 1,000 frames
    - Reduces moving-target instability
- **Epsilon-Greedy Exploration:**

- Epsilon decays from 1.0 → 0.01 over 150,000 frames
- Ensures adequate exploration before exploitation

Optimization

- Optimizer: Adam
- Learning rate: 1e-4
- Gamma (discount): 0.99
- Batch size: 32

This setup forms a representative and stable DQN implementation.

Double DQN (DDQN) Variant

Double DQN addresses a known problem in DQN: overestimation bias.
Standard DQN selects and evaluates the best action using the same target network:

Double DQN separates these:

- **Online network** selects the best next action
- **Target network** evaluates it

This modification reduces inflated Q-values and typically leads to smoother learning curves and better stability.

In my implementation, DDQN is controlled by a flag:

USE_DOUBLE_DQN = True

Experimental Setup

Hyperparameters

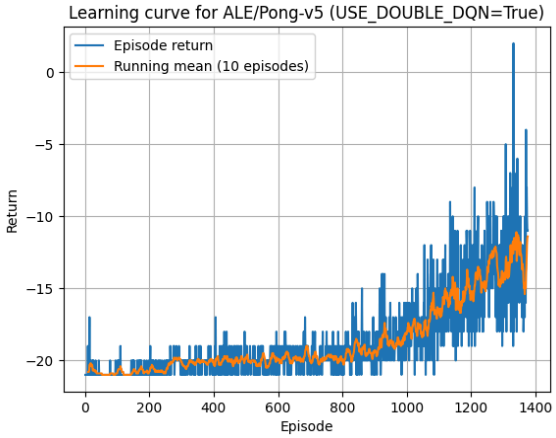
Parameter	Baseline DQN	Double DQN

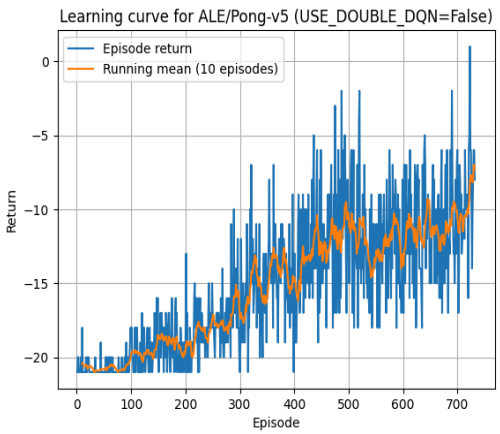
Learning rate	1e-4	1e-4
Discount γ	0.99	0.99
Replay size	10,000	10,000
Batch size	32	32
Target sync	1,000 frames	1,000 frames
Epsilon decay	150k frames	150k frames
Frame stack	4	4

Training Procedure

- Train for up to 500,000 frames (Colab-friendly cap)
- Log episode rewards
- Generate:
    - Early/random video
    - Learned/late video
- Repeat for both baseline + DDQN

Results





General trends:

- **Baseline DQN**
    - The learning curve fluctuates significantly.
    - Shows signs of overestimation bias (large jumps in return).
    - Eventually improves but with notable instability.
- **Double DQN**
    - Much smoother learning curve.
    - Reaches positive returns earlier.
    - More stable Q-value updates.

Gameplay Behavior

Early Policy

- The paddle moves randomly.
- No consistent tracking of the ball.
- Frequent misses.

Learned Policy

- Tracks the ball reliably.
- Positions paddle early based on ball velocity.
- Sustains longer volleys and accumulates positive scores.

Reflection

For this project, I selected Atari Pong because it is one of the most established benchmarks for value-based reinforcement learning and provides a clear environment for analyzing how architectural and algorithmic choices affect agent behavior. Pong has a simple discrete action space and a predictable yet non-trivial reward structure, making it ideal for studying learning dynamics in both a baseline DQN and an improved variant. “Improvement” in this context meant seeing the agent progress from random paddle movement and near-zero performance to purposeful tracking of the ball, more consistent volleys, and increasingly positive episodic returns. The visual transition between the early and learned gameplay videos clearly demonstrated this shift, initially chaotic behavior gradually became disciplined and reactive as the agent learned.

A major challenge during training was the sparse and delayed reward signal. The agent only receives rewards when scoring or losing a point, which happens relatively infrequently. This caused early episodes to provide little learning signal, making effective exploration essential. The epsilon-greedy exploration schedule and replay buffer helped the agent break out of the cold start phase, while the target network stabilized training by preventing harmful feedback loops. Another challenge was the Q-value overestimation bias characteristic of standard DQN. This often led to unstable value updates and noisy learning curves. Switching to Double DQN significantly mitigated this problem by separating action selection (online network) from action evaluation (target network). This resulted in smoother learning curves, more stable convergence, and noticeably stronger gameplay in the learned video.

If I continued this project, I would explore more advanced techniques to further improve stability and data efficiency. Prioritized Experience Replay (PER) could help the agent focus on more informative transitions, especially in sparse-reward environments like Pong. N-step returns could improve credit assignment by propagating rewards more effectively through long sequences with zero reward. I would also experiment with different epsilon decay schedules, target network update intervals, and sticky actions, which add environmental stochasticity and often lead to more robust policies. Additionally, the Dueling DQN architecture could help the network separately estimate state value and action advantage, potentially improving learning speed. Overall, this project provided valuable insight into the strengths and limitations of value-based deep RL and demonstrated how even relatively small algorithmic modifications can lead to noticeably improved performance.

Conclusion

This project successfully demonstrated:

- Implementation of a full baseline DQN for Atari Pong
- Implementation of a Double DQN variant

- Comparison of training stability and performance
- Video capture showing qualitative improvement

Double DQN provided clearer and more stable learning, validating its advantage over standard DQN in reducing overestimation bias.

Future work could include Prioritized Experience Replay, N-step returns, or Dueling Networks, which may further improve data efficiency and robustness.

8. References

- Mnih et al., **“**Human-level control through deep reinforcement learning.” *Nature*, 2015.
- Van Hasselt et al., *“Deep Reinforcement Learning with Double Q-Learning.” AAAI, 2016.*