Deep Reinforcement Learning for Atari Breakout

William Riser
riser.w@northeastern.edu
CS 4100 – Artificial Intelligence, Northeastern University

Spring 2025

Abstract

This work investigates applying Deep Q-Networks (DQN) to the Atari Breakout environment using the 128-byte RAM state. We outline the experimental framework, architectural choices, and training protocol at a high level. Our results show how network design and hyperparameter settings impact learning stability and efficiency. We present the benefits such as training speed while also highlighting some of the drawbacks of following this approach. Additionally, we present potential modifications that have the potential to greatly improve the performance of this method.

1 Introduction

Deep reinforcement learning (RL) has enabled agents to learn complex behaviors directly from high-dimensional inputs by combining value-based methods with deep neural networks [1]. The Atari Learning Environment (ALE) provides a benchmark suite where agents must map raw observations to control policies [2]. While most prior work focuses on pixel-based inputs, the 128-byte RAM state offers a compact, lossless representation of game dynamics that can substantially reduce sample complexity and training time [7]. In this work, we implement the Deep Q-Network (DQN) algorithm in Gymnasium's 'Breakout-ram-v5' environment to investigate how architectural and hyperparameter choices impact learning stability and efficiency when using RAM states. We provide a comparison of key design decisions, such as network depth, target update frequency, and replay buffer sizing, in the context of RAM-based DQN. We also discuss practical implications for rapid prototyping of RL agents, and outline extensions such as Double DQN [5] and prioritized replay that may further reduce value overestimation and improve sample efficiency.

2 Background and Related Work

The Atari Breakout task can be described as a Markov Decision Process (MDP) $\langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$, where the agent interacts with the Arcade Learning Environment (ALE) to maximize long-term score.

State Space $\mathcal{S}$. Each state s_t is represented by a 128-dimensional RAM vector, $s_t \in \{0, \dots, 255\}^{128}$, capturing the complete internal memory of the game at time t. This compact encoding preserves all relevant information such as paddle and ball positions, remaining bricks, and life count while dramatically reducing input dimensionality compared to raw pixel observations.

Action Space $\mathcal{A}$. At each time step the agent selects one of four discrete controls:

$$\mathcal{A} = \{\texttt{NOOP}, \texttt{FIRE}, \texttt{RIGHT}, \texttt{LEFT}\},$$

corresponding to no operation, launching the ball, and horizontal paddle movements.

Transition Dynamics $P(s' \mid s, a)$. Transitions are provided deterministically from the ALE emulator's internal mechanics. Given a current RAM state s and action a, the environment advances one frame to a new state s'. Although these dynamics are fully observable through the RAM vector, they are not modeled and must be learned through agent–environment interaction.

Reward Function $R(s, a, s')$. The agent receives the game's raw score increment as its reward:

$$R(s, a, s') = \begin{cases} 1\text{–}4, & \text{if a brick is broken in } (s \to s'), \\ 0, & \text{otherwise.} \end{cases}$$

This immediate reward structure encourages the agent to clear bricks efficiently.

Discount Factor and Objective. Future rewards are discounted by $\gamma = 0.999$, emphasizing long-horizon planning. The objective is to find a policy $\pi(a \mid s)$ that maximizes the expected return

$$Q(\pi) = E_\pi \Big[\sum_{t=0}^{\infty} \gamma^t \, r_t \Big],$$

where $r_t = R(s_t, a_t, s_{t+1})$.

Initialization and Termination. Episodes begin from an initial state corresponding to the standard Breakout start screen. An episode terminates when the agent loses all lives or after a fixed maximum length of 1,000 steps, whichever occurs first.

Deep Q-Networks Deep Q-Networks (DQN) approximate $Q(s, a; \theta)$ with a parameterized neural network, employing experience replay to decorrelate updates and a target network to stabilize training [1]. Extensions such as Double DQN have been proposed to reduce overestimation bias by decoupling action selection and evaluation [5]. Many of the current studies concentrate on agents consuming raw pixel inputs; significantly fewer studies have explored DQN on RAM state representations, despite their advantage in reducing input size. Our study evaluates how standard DQN enhancements transfer to the RAM domain.

3 Methodology

3.1 Experimental Framework

We conduct experiments within the ALE 'Breakout-ram-v5' environment using Gymnasium [7]. Agents interact over 35,000 episodes, with checkpoints saved every 1,000 episodes for offline analysis.

3.2 State Representation

The agent receives a 128-dimensional RAM vector, which reduces input complexity compared to raw RGB frames while retaining critical game state information. Specifically, RAM state representation uses 128 bytes of memory compared to 100,800 bytes in the RGB variant.

3.3 Network Architecture

The Q-network consists of three fully connected layers with ReLU activations, mapping from the 128-dimensional input to four action-value outputs. This architecture balances representational capacity with computational efficiency. Through experimentation, we found that adding layers provided no additional performance while significantly slowing down training.

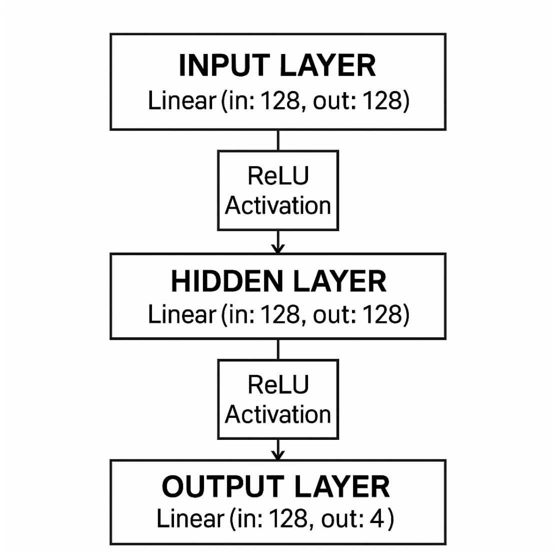

Figure 1: Neural network architecture

3.4 Training Protocol

Agents are trained using an epsilon-greedy exploration strategy, decaying ϵ from 1.0 to 0.01. The loss function is the mean squared error between current Q-values and TD targets:

$$y_t = r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \theta^-),$$

with θ^- denoting target network parameters. Target networks are synchronized every 4 training steps to improve stability [1].

3.5 Hyperparameter Selection

Hyperparameters were chosen based on prior reports and preliminary tuning to balance learning speed and stability:

Table 1: Key Hyperparameters

Hyperparameter	Value
Replay buffer size	50,000
Mini-batch size	64
Discount factor (γ)	0.999
Learning rate (Adam)	1×10^{-4}
Target update frequency	4 steps
ϵ-decay	0.999
Max episode length	1,000 steps

4 Experimental Setup

Each episode terminates upon game completion or after 1,000 steps. Performance is evaluated using episode rewards and a 100-episode rolling average.

5 Discussion

Our investigation reveals that RAM-based DQN can achieve competitive learning dynamics with reduced computational overhead. Early-stage training exhibits high reward variance, which stabilizes as ϵ decays. While performance plateaus below published pixel-based benchmarks [1], the simplicity of the state representation offers advantages for rapid prototyping and experimentation. The agent reliably tracks and returns the ball, however it never learns the "tunneling" tactic which occurs when creating a gap in the brick wall and steering the ball through it so that it bounces behind the bricks, clearing them rapidly without frequent paddle hits.

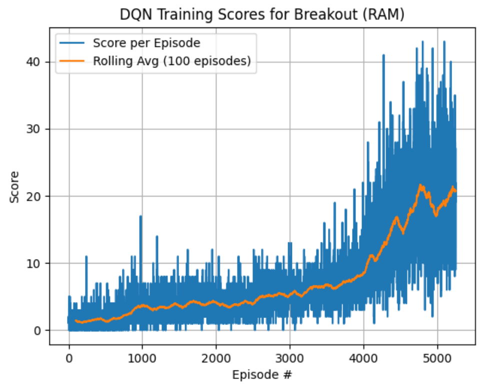

Figure 2: Average score graph during training

5.1 Challenges

Despite the streamlined setup, several challenges emerged during development and experimentation:

- **High Variance in Early Training.** The combination of sparse rewards and an ϵ-greedy policy leads to widely fluctuating episode returns in the first few thousand episodes. This makes it difficult to gauge whether hyperparameter adjustments yield genuine improvement or are a result of noise.

- **Hyperparameter Sensitivity.** Learning rate and target-network update frequency strongly influence convergence behavior. Infrequent updates (every 10+ steps) slowed error correction, whereas overly frequent updates (every step) undermined the stabilizing effect of the target network and significantly slowed training time.

- **Overestimation Bias.** Standard DQNs use of the max operator in the TD target can overestimate action values, leading to suboptimal policies.
- **Replay Buffer Limitations.** A fixed-size buffer can discard important transitions. Uniform sampling further dilutes the impact of these important experiences on gradient updates.
- **Computational Constraints.** Although RAM-based inputs reduce per-step computation, training over tens of thousands of episodes still demands significant compute resources.
- **Partial Observability.** Even though the RAM state contains all internal variables, it can be challenging to disentangle the game dynamics (e.g., ball direction and velocity) without manual feature engineering or deeper architectures.

6 Future Work

While our RAM-based DQN implementation offers rapid prototyping advantages, several methods remain to further enhance performance and generality.

- **Double DQN and Dueling Architectures.** Integrate Double DQN [5] to mitigate value overestimation bias, and experiment with the dueling network architecture [8] to separately estimate state-value and advantage functions, which has been shown to accelerate learning in Atari domains.
- **Prioritized Experience Replay.** Replace uniform sampling with prioritized replay [9] to focus updates on more important transitions, potentially improving sample efficiency and reducing convergence time.
- **Multi-Step Targets.** Incorporate multi-step returns to propagate rewards more quickly, and evaluate distributional RL methods to model the full return distribution rather than just its expectation [10].
- **Transfer Across Games.** Extend the framework to additional ALE titles (e.g. Space Invaders, Pong) to assess how RAM-based agents generalize across diverse game mechanics and environments.
- **Pixel-Based and Hybrid Inputs.** Compare RAM-only agents against RGB-based CNN architectures, or hybrid models that combine RAM and RGB inputs, to evaluate the trade-off between input size and final performance.

7 Conclusion

We have presented a streamlined implementation of DQN using the 128-byte RAM state for Atari Breakout, focusing on methodological clarity and experimental reproducibility. Our results demonstrate that while RAM-based agents learn more rapidly and require lower computational overhead than the pixel-based alternative, they ultimately plateau at lower performance levels. This demonstrates the value of visual features for fine-grained control in high-dimensional tasks. The compact state representation, however, offers a compelling platform for rapid prototyping. Future work will integrate Double DQN and prioritized experience replay to further stabilize learning, and extend the comparative framework to other ALE titles to assess generality across diverse game mechanics.

References

[1] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.

[2] M. G. Bellemare *et al.*, "The Arcade Learning Environment: An evaluation platform for general agents," *J. Artif. Intell. Res.*, vol. 47, pp. 253–279, 2013.

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed., MIT Press, 2018.

[4] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[5] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI*, 2016, pp. 2094–2100.

[6] G. Brockman *et al.*, "Gymnasium: A toolkit for developing and comparing reinforcement learning algorithms," *arXiv preprint arXiv:1606.01540*, 2016.

[7] M. Towers *et al. Gymnasium: A Standard Interface for Reinforcement Learning Environments.* arXiv preprint arXiv:2407.17032, 2024. Available at: `https://arxiv.org/abs/2407.17032`

[8] Z. Wang *et al.*, "Dueling Network Architectures for Deep Reinforcement Learning," arXiv preprint arXiv:1511.06581, 2016. Available at: `https://arxiv.org/abs/1511.06581`

[9] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized Experience Replay," arXiv preprint arXiv:1511.05952, 2016. Available at: `https://arxiv.org/abs/1511.05952`

[10] M. G. Bellemare, W. Dabney, and R. Munos, "A Distributional Perspective on Reinforcement Learning," arXiv preprint arXiv:1707.06887, 2017. Available at: `https://arxiv.org/abs/1707.06887`