Deep Q-Network (DQN) Training and Hyperparameter Analysis

Jaiv Chaitanya Burman

---

1. Introduction

Deep Q-Networks (DQN) have become a fundamental approach to reinforcement learning by integrating deep neural networks with Q-learning. This report explores how different **batch sizes** and **target update rates** impact the training performance of a DQN agent.

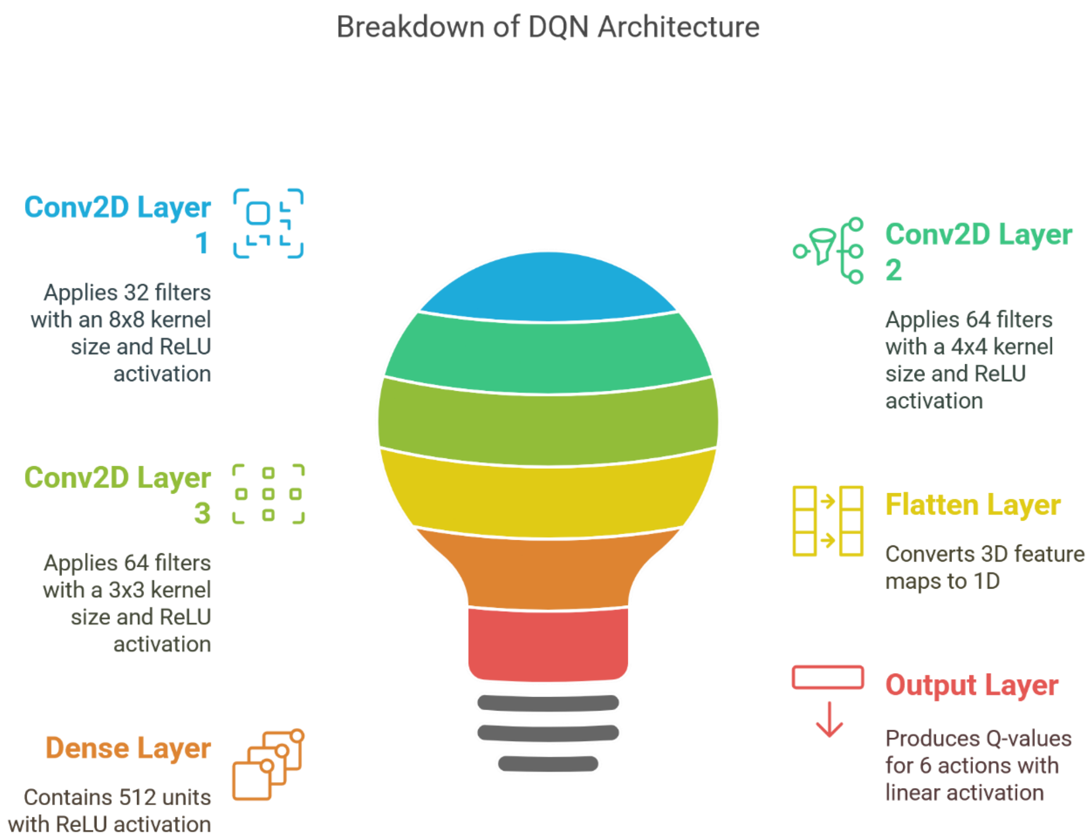
We experiment with four different hyperparameter settings:

1. **Batch = 8, Target Update = 10**
2. **Batch = 16, Target Update = 10**
3. **Batch = 8, Target Update = 3**
4. **Batch = 16, Target Update = 3**

The goal is to determine which setting provides the best balance between **learning stability and performance**.

2. Network Architecture

The Deep Q-Network (DQN) used in this experiment follows a **convolutional neural network (CNN) architecture** to process visual input from the environment.



DQN Model Summary

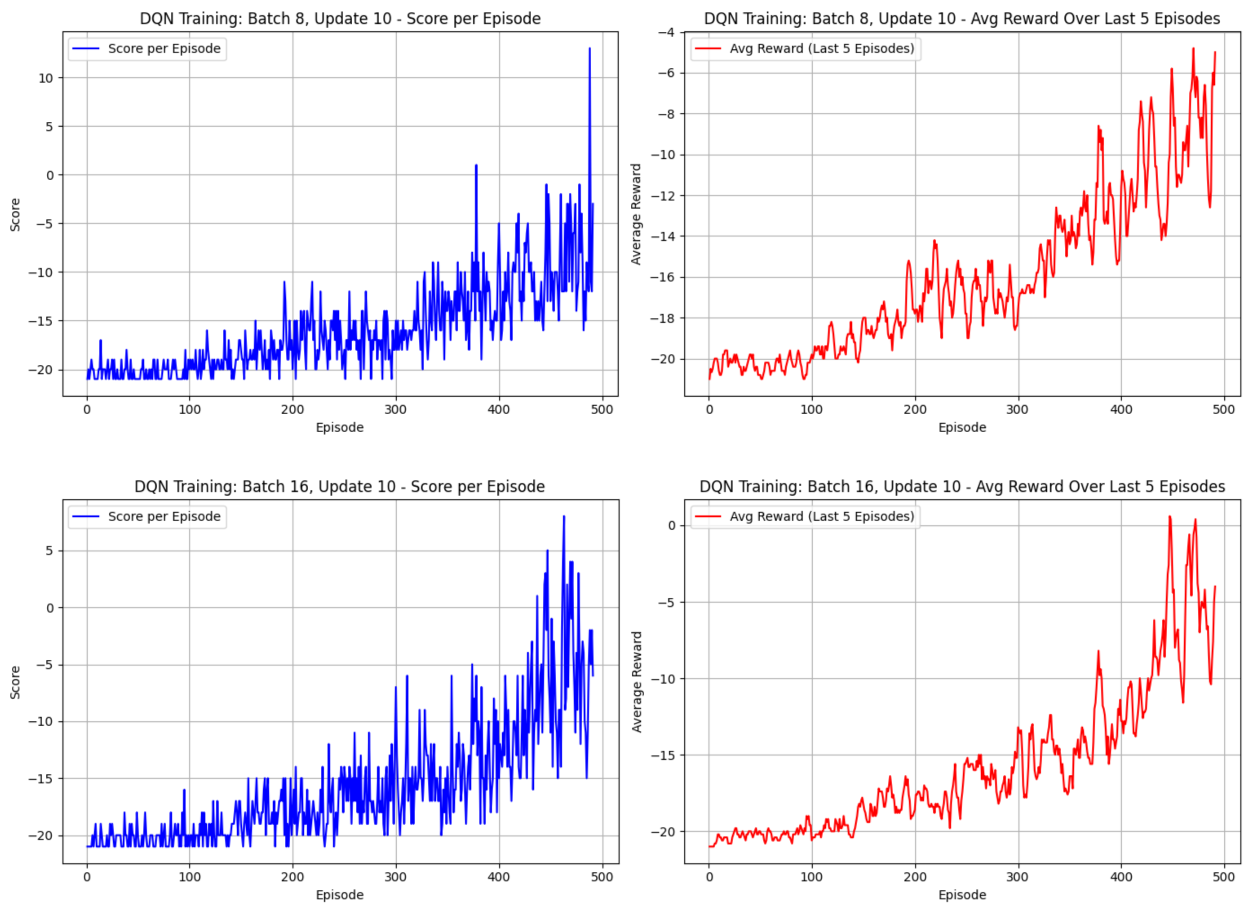
Layer Type	Filters/Units	Kernel Size	Activation
Conv2D	32	(8,8)	ReLU
Conv2D	64	(4,4)	ReLU
Conv2D	64	(3,3)	ReLU
Flatten	-	-	-
Dense	512	-	ReLU
Output Layer	6 (Actions)	-	Linear

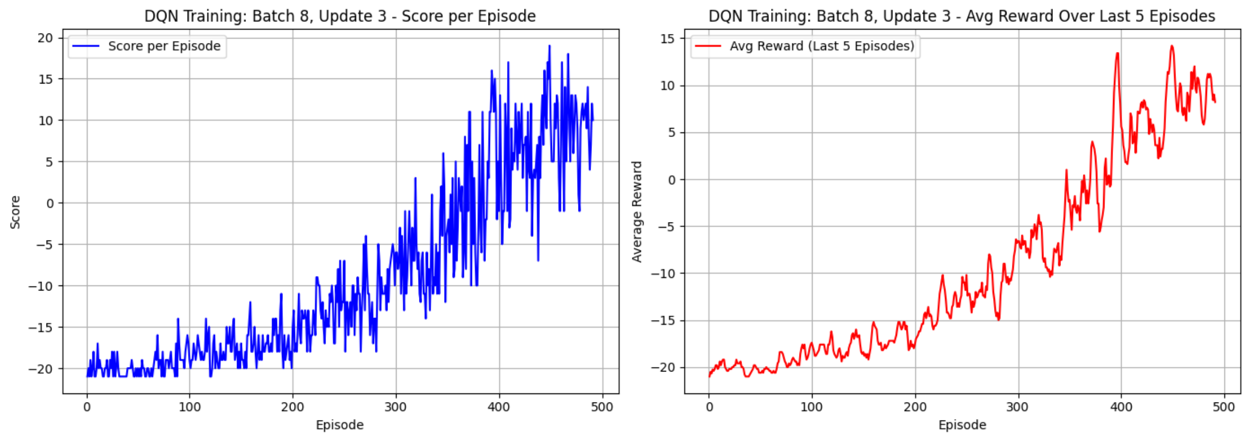
- **Input Shape:** (4, 84, 80) → A stacked frame of 4 images from the environment.
- **Output:** Q-values for **6 possible actions** in the environment.
- **Optimization Algorithm:** Adam (lr = 0.00025).
- **Loss Function:** Mean Squared Error (MSE).

3. Training Metrics and Observations

We trained the DQN agent with different **batch sizes** and **target update frequencies** to analyze how these parameters affect learning.

The following plot shows how the **total score & Average Reward Over last 5 episodes** evolves over training:





- **Observations:**
    - Smaller **batch sizes (8)** seem to result in **more fluctuations** in scores.
    - **Batch = 16** provides **smoother learning curves** with fewer spikes.
    - **Target update = 10** leads to **better long-term performance**, while target update = 3 seems too frequent and introduces instability.

Average Reward Over Last 5 Episodes

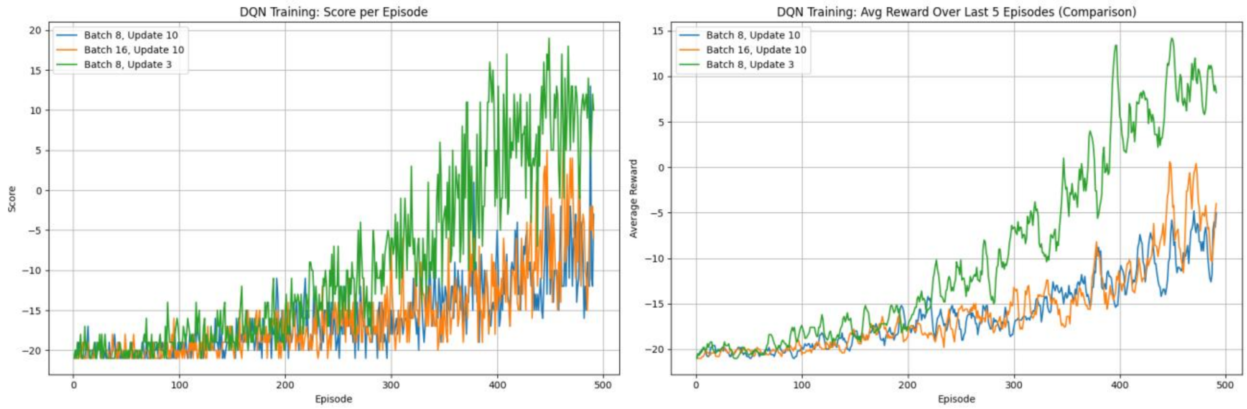
To assess **stability**, we track the **moving average reward** over 5 episodes:

Insert average reward graph here

- **Observations:**
    - **Batch = 8, Update = 3** achieves the **highest average reward**, indicating faster learning.
    - **Batch = 16, Update = 10** produces a **more stable learning curve**, though it converges slightly slower.

Combined Comparisons

We compare **all four configurations** in one plot:



- **Observations:**
    - **Frequent target updates (Update = 3) lead to higher variance**.
    - **Larger batch size (Batch = 16) results in more consistent learning**.
    - **Best performing model:** Batch = 8, Target Update = 3 (highest reward growth).

4. Best Hyperparameter Choice

After analyzing the results, the best configuration depends on the desired outcome:

- **For stability:** Batch = 16, Target Update = 10 (consistent learning, low variance).
- **For fast learning:** Batch = 8, Target Update = 3 (highest average reward, but more variance).

Final Choice:
If we prioritize faster learning and higher rewards, ***Batch = 8, Target Update = 3*** *is the best choice.*

5. Conclusion

This study highlights the **trade-offs** between batch size and target network update frequency in DQN training:

- **Larger batch sizes** lead to **smoother learning** but can slow convergence.
- **Frequent target network updates** (update = 3) improve adaptation but **increase variance**.
- The **best trade-off** is using Batch = 8 with Target Update = 3, which **achieves the highest rewards while maintaining reasonable stability**.