Assignment 1: Q-Learning Tabular and Deep

Benard Adala Wanyande [1]

Abstract

Reinforcement Learning (RL) has shown significant advancements in solving complex decision-making problems through trial-and-error interactions with an environment. Traditional tabular Q-learning struggles with scalability when the state-action space is large (cannot fit in memory), necessitating the use of function approximation. This work explores Deep Q-Networks (DQN) as a function approximator to enhance learning efficiency, particularly in the CartPole environment. We analyze the **derivation of the DQN loss function**, compare it with tabular Q-learning updates, and **investigate the role of target networks** and **experience replay** in stabilizing deep q-learning. Our experiments evaluate the effect of key hyperparameters and demonstrate how deep function approximation improves RL performance. The findings emphasize the necessity of stability techniques in deep RL and provide insights into optimizing Q-learning in large state spaces.

1. Introduction

Reinforcement Learning (RL) enables agents to learn optimal decision-making policies through interaction with an environment, receiving feedback in the form of rewards (Sutton & Barto, 2018). While tabular Q-learning effectively finds optimal policies for small state spaces, it becomes impractical for high-dimensional or continuous environments due to the **curse of dimensionality** (Mnih et al., 2015). To address this issue, Deep Q-Networks (DQN) use neural networks as function approximators, allowing RL agents to generalize across large state spaces (Mnih et al., 2013; 2015). However, deep RL introduces **instability** and **convergence challenges**, requiring techniques such as **experience replay** (Lin, 1992; Mnih et al., 2015) and **target networks** (Mnih et al., 2015; van Hasselt et al., 2016b) to improve learning stability.

[1]Leiden University, Leiden, Netherlands. Correspondence to: Benard Adala Wanyande <b.a.wanyande@umail.leidenuniv.nl>.

In this work, we implement and analyze deep Q-learning in the CartPole environment (Brockman et al., 2016), comparing naive function approximation with stability-enhancing techniques, and conducting an ablation study to examine the impact of hyperparameters. Through this, we aim to understand how deep RL can effectively scale and improve learning efficiency in complex environments.

2. From Tabular Q-Learning to Deep Q-Learning

2.1. Q-Learning Update Rule

Q-learning is a model-free reinforcement learning algorithm that updates action-value estimates iteratively using the Bellman equation (Watkins & Dayan, 1992; Sutton & Barto, 2018):

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right], \tag{1}$$

where:

- $Q(s,a)$ represents the current estimate of the action-value function.
- r is the reward received after taking action a in state s.
- s' is the next state.
- γ is the discount factor, determining the importance of future rewards.
- α is the learning rate, controlling update intensity.

This update refines the Q-values by minimizing the difference between the estimated Q-value and the temporal difference (TD) target (Sutton & Barto, 2018).

2.2. Transition to Function Approximation

In large or continuous state spaces, maintaining a tabular representation of $Q(s,a)$ is infeasible due to memory constraints (Sutton & Barto, 2018). This is why we cannot use update rules as in tabular RL.

Instead, function approximation is employed, where a neural network parameterized by weights θ approximates $Q(s, a)$:

$$Q(s, a; \theta) \approx Q(s, a). \tag{2}$$

Deep Q-Networks (DQN) extend Q-learning by using deep neural networks to approximate Q-values, enabling learning in complex environments (Mnih et al., 2015).

2.3. Derivation of the DQN Loss Function

Since neural networks are optimized using gradient descent, we require a differentiable loss function. The TD target for Q-learning is:

$$y_t = r + \gamma \max_{a'} Q(s', a'; \theta^-), \tag{3}$$

where θ^- represents the parameters of a separate target network to stabilize learning. The **loss function for DQN** is formulated as the **Mean Squared Error (MSE)** between the **TD target** and the **predicted Q-value** (Mnih et al., 2015):

$$\mathcal{L}(\theta) = \mathbb{E}\left[(y_t - Q(s, a; \theta))^2\right]. \tag{4}$$

2.4. DQN with Experience Replay and Target Networks

Deep Q-Networks (DQN) extend tabular Q-learning by using a neural network to approximate $Q(s, a)$, making reinforcement learning scalable to large state spaces (Mnih et al., 2015). However, deep Q-learning introduces **instability and divergence issues**, primarily due to:

1. **Moving Target Problem**: The Q-network is updated while also being used to generate targets, causing unstable updates (van Hasselt et al., 2016a).
2. **Strong Temporal Correlations**: Consecutive experiences in an episode are highly correlated, leading to inefficient learning (Sutton & Barto, 2018).

To mitigate these issues, **experience replay (ER)** and **target networks (TN)** are introduced (Mnih et al., 2015).

The experience replay approach stores past experiences (s, a, r, s') in a replay buffer, allowing the model to sample mini-batches randomly. This breaks the correlation between consecutive samples and improves data efficiency.

The target networks approach uses a separate, periodically updated network to generate stable target values. This prevents the model from chasing moving targets during updates (van Hasselt et al., 2016a).

2.4.1. Pseudocode for DQN with Experience Replay and Target Networks

Algorithm 1 Deep Q-Network (DQN)

```
Initialize empty replay buffer
Initialize Q-network with random weights
Initialize target network Q_t with random weights
Set initial state s = s_0
while not converged do
    Select action a using ε-greedy policy
    Execute a, observe reward r and next state s'
    Store (s, a, r, s') in replay buffer
    Sample mini-batch from replay buffer
    Compute TD target using target network Q_t
    Update Q-network via gradient descent
    Periodically update target network: Q_t ← Q
end while
```

3. Experiments

3.1. Experiment Setup

This study evaluates the performance of Deep Q-Learning (DQN) in the CartPole environment, analyzing the effects of hyperparameters and stability techniques such as target networks and experience replay. The environment used is **CartPole-v1**, where the agent must balance a pole on a moving cart while maximizing cumulative reward. A **fully connected neural network** approximates the Q-function, trained using **mini-batch gradient descent** with an **epsilon-greedy policy** for action selection. The key metric is **return**, the cumulative reward per episode, tracked over training steps to assess convergence.

3.2. Experimental Methodology

Each experiment was repeated across **five independent runs** to ensure reliability, with each run consisting of **1,000,000 environment steps**. The agent's performance was measured using the **mean episode return**, while the **standard deviation** was visualized as a confidence interval in learning curves to account for variability in training.

3.3. Experimental Objectives

The study investigates the impact of key hyperparameters on learning performance. The learning rate determines the step size of weight updates, with lower values promoting stability but slowing convergence, while higher values accelerate learning but may cause instability. The discount factor influences the agent's long-term planning, where lower values prioritize immediate rewards, while higher values favor future returns. The exploration rate controls the balance between exploration and exploitation, with slower decay maintaining exploration for longer periods, whereas faster decay leads to early exploitation. The neural network architecture affects function approximation, with larger networks

providing more representational capacity but increasing the risk of overfitting.

To further assess training efficiency, different DQN configurations were compared. The baseline Naive DQN was implemented without stabilization mechanisms. Target networks were introduced to stabilize updates by maintaining a slowly updated copy of the Q-network. Experience replay was tested to improve sample efficiency and decorrelate training data. The combination of target networks and experience replay was examined to determine whether integrating both techniques enhances learning stability and policy robustness.

3.4. Expected Outcomes

A well-configured Deep Q-Network (DQN) should show steady performance improvements over time, with optimal hyperparameter choices accelerating convergence while ensuring stability (Mnih et al., 2015). Stability techniques such as target networks and experience replay mitigate divergence issues, with target networks reducing oscillations in value estimates and experience replay improving sample efficiency by decorrelating training updates (van Hasselt et al., 2016a). The combination of both techniques is expected to yield the most stable and efficient learning (Zhang & Sutton, 2017).

This study first examines the effect of hyperparameters in naive DQN, followed by an evaluation of stabilization techniques to assess their role in improving training consistency and policy robustness.

3.5. Ablation Study on Hyperparameters

Understanding the role of hyperparameters in Deep Q-Network (DQN) training is essential for developing a stable and effective agent. This ablation study examines the effects of learning rate, discount factor, exploration rate, and network size on training performance. Each experiment was conducted over five independent runs, with 1,000,000 training steps per run. The results were averaged, and standard deviation was used to account for variability.

3.5.1. Effect of Learning Rate

The learning rate determines how aggressively the network updates its parameters. A low learning rate of 0.0001 resulted in slow learning, requiring significantly more steps to improve performance. Increasing the rate to 0.0005 provided a balanced trade-off between stability and speed. A higher rate of 0.001 led to faster convergence but also introduced slightly greater variance. The most effective setting was 0.001, which allowed the agent to reach the highest rewards in the shortest time.

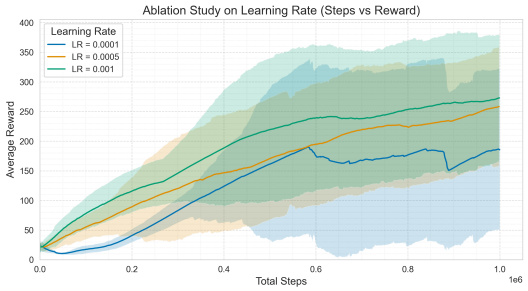

Figure 1. Ablation Study on Learning Rate

3.5.2. Effect of Discount Factor (Gamma)

The discount factor influences how much the agent values future rewards. A lower gamma of 0.85 led to short-sighted behavior, where the agent prioritized immediate gains. Increasing gamma to 0.95 allowed for more balanced decision-making. The highest gamma value of 0.99 resulted in the best performance, enabling the agent to maximize future rewards and sustain longer episodes. Higher gamma settings encouraged more strategic long-term planning.

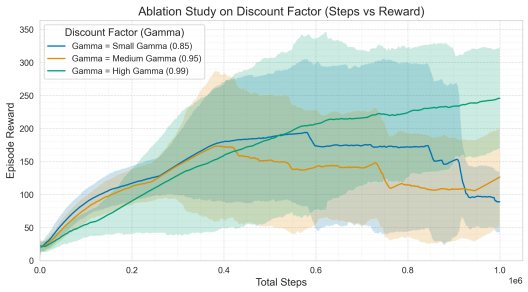

Figure 2. Ablation Study on Discount Factor

3.5.3. Effect of Exploration Rate (Epsilon)

Exploration determines how frequently the agent tries new actions instead of exploiting known strategies. A high exploration setting ($\epsilon_{\text{decay}} = 0.9999$) led to slow convergence, as the agent continued exploring long after effective policies had been discovered. Reducing exploration too quickly ($\epsilon_{\text{decay}} = 0.99$) caused premature exploitation, resulting in suboptimal policies. A balanced approach with a decay of 0.999 provided the best results, ensuring adequate exploration while transitioning smoothly to exploitation.

3.5.4. Effect of Neural Network Size

The neural network's complexity affects function approximation. A small network with one hidden layer of 32

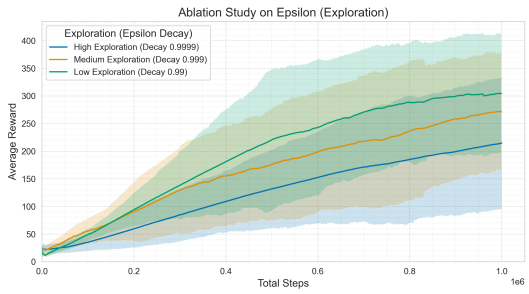

Figure 3. Ablation Study on Exploration Rate

neurons struggled to capture the complexity of the environment, leading to slow convergence and limited performance. A larger network with three layers of 128 neurons converged quickly but exhibited high variance and signs of overfitting. The medium-sized network, consisting of two layers with 64 neurons each, demonstrated the best balance between stability and performance, making it the most effective choice.

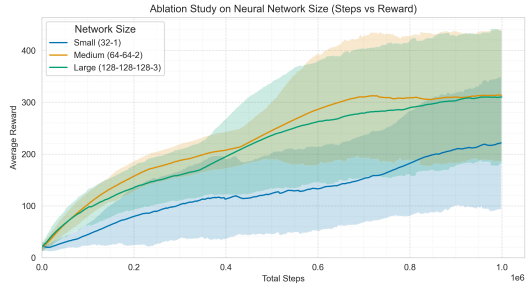

Figure 4. Ablation Study on Neural Network Size

3.5.5. Summary of Hyperparameter Findings

The best-performing configuration for DQN training was a learning rate of 0.001, a discount factor of 0.99, an epsilon decay of 0.999, and a neural network with two hidden layers of 64 neurons each. These settings provided the most stable and efficient training results. The next stage of experimentation explores stabilization techniques such as target networks and experience replay to further enhance training performance.

3.5.6. Best Hyperparameter Configuration

Following extensive experimentation, the best hyperparameter configuration for training the DQN agent was determined based on stability, convergence speed, and final performance. The optimal **learning rate** was found to be $\alpha = 0.001$, providing fast convergence without instability. The **discount factor** $\gamma = 0.99$ enabled effective long-term planning while minimizing excessive variance. The most effective **exploration decay rate** was $\epsilon_{\text{decay}} = 0.999$, balancing early exploration with later exploitation. A **medium network architecture** consisting of two hidden layers with 64 neurons each (64-64-2) provided the best trade-off between function approximation and training efficiency.

This optimal configuration ensured that the agent consistently reached high episode returns within the allocated training budget while maintaining generalization across different runs. With these hyperparameters, the agent achieved a reward plateau above 200 within significantly fewer training steps compared to suboptimal settings. Despite these improvements, instability in training remained a challenge, necessitating further evaluation of stabilization techniques.

3.6. Ablation Study on Stabilization Techniques

While hyperparameter tuning improved performance, the agent still exhibited variability in training due to bootstrapping errors and function approximation instability. To address these issues, this second ablation study focused on evaluating the impact of stabilization techniques. The effectiveness of target networks, experience replay, and their combination was examined to determine whether they enhanced training consistency and final policy performance.

3.6.1. Effect of Experience Replay

Experience replay was introduced by storing past transitions in a fixed-size buffer and sampling mini-batches for training. This technique aimed to decorrelate consecutive updates, reducing instability caused by highly correlated data. Results indicated that experience replay improved training efficiency, leading to smoother learning curves and better generalization. The agent reached optimal rewards more consistently across runs, though early-stage training was slower due to the time required to populate the buffer.

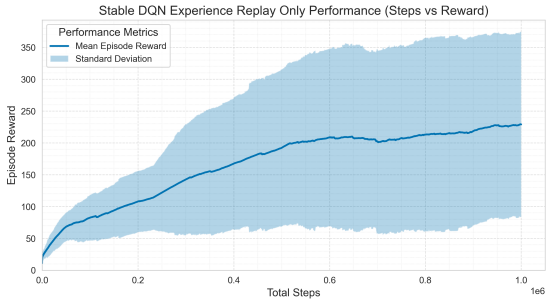

Figure 5. Impact of Experience Replay on Training Stability

3.6.2. Effect of Target Networks

A target network was implemented to stabilize updates by maintaining a slowly updated copy of the Q-network. This prevented rapid oscillations in value estimates, addressing divergence issues. The introduction of a target network resulted in a more controlled training process, with reduced variance across independent runs. However, without experience replay, sample efficiency remained a challenge, leading to slower initial learning.

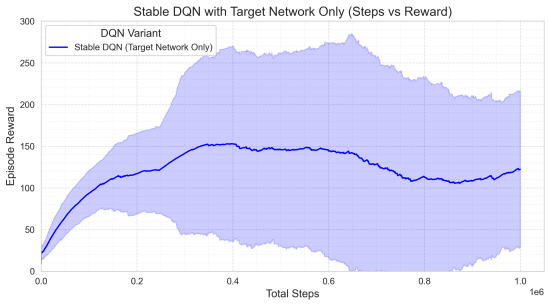

Figure 6. Impact of Target Networks on Training Stability

3.6.3. Effect of Combining Target Networks and Experience Replay

The combination of target networks and experience replay yielded the most stable learning dynamics. Experience replay improved sample efficiency by breaking temporal correlations, while the target network ensured that value updates remained stable. This configuration resulted in the smoothest learning curves and the highest final episode returns, demonstrating that stabilization techniques mitigate function approximation issues inherent to DQN.

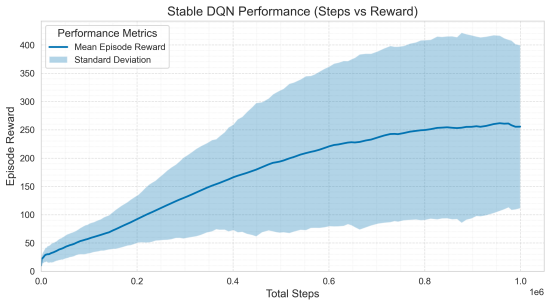

Figure 7. Impact of Target Networks and Experience Replay Combined

3.6.4. Summary of Stabilization Findings

The most effective configuration incorporated both target networks and experience replay, producing the most stable and generalizable policy. While experience replay alone improved sample efficiency and target networks reduced variance, their combination provided the best results. These findings highlight the necessity of stabilization techniques for deep reinforcement learning, particularly in environments susceptible to function approximation errors.

The final analysis suggests that deep Q-learning, when optimized with effective hyperparameters and stability techniques, can successfully learn robust policies. Future work may explore additional enhancements such as prioritized experience replay or dueling architectures to further refine training efficiency and policy performance.

4. Discussion

4.1. Results

The experiments demonstrated the impact of hyperparameters and stabilization techniques on the performance of Deep Q-Networks (DQN) in the CartPole environment. The first ablation study focused on understanding the sensitivity of DQN to key hyperparameters, while the second study investigated the role of stabilization methods in improving training consistency and generalization.

Hyperparameter tuning revealed that a high learning rate of 0.001 accelerated convergence without causing instability, while a high discount factor of 0.99 encouraged long-term reward maximization. A moderate exploration decay rate of 0.999 provided a balance between exploration and exploitation, leading to efficient policy learning. The medium network size (64-64-2) offered the best trade-off between computational efficiency and function approximation capability.

The stabilization study demonstrated that both target networks and experience replay contributed to improved training stability. Experience replay enhanced sample efficiency by reducing correlation between updates, while target networks prevented divergence by stabilizing Q-value estimates. The combination of both methods resulted in the smoothest learning curves, minimal variance across independent runs, and the most robust final policy.

Figure 8 provides a comparative analysis of the different DQN configurations. The naive DQN (blue) exhibits slow convergence, plateauing at a reward of approximately 200 with high variance. The introduction of experience replay (orange) significantly enhances stability, allowing the agent to achieve higher episode returns with smoother learning curves. The use of a target network alone (green) stabilizes value estimates but slows overall learning due to delayed

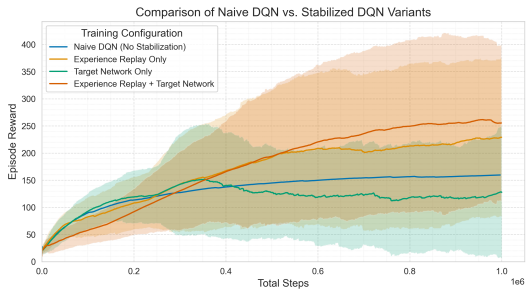

Figure 8. Comparison of Naive DQN vs. Stabilized DQN Variants

target updates. The most effective configuration, combining both experience replay and target networks (red), demonstrates the highest convergence speed, lowest variance, and best final performance.

A summary of these findings is presented in Table 1, which highlights the impact of each stabilization technique on convergence speed, final performance, and training stability.

Configuration	Convergence Speed	Final Performance	Stability
Naive DQN (No Stability)	Slow	Moderate (Plateau at 200)	High Variance
Experience Replay Only	Moderate	High	Improved Stability
Target Network Only	Slow	High	Reduced Variance
Both (Target + Replay)	Fastest	Highest	Most Stable

Table 1. Comparison of different DQN stabilization techniques

These results confirm that naïve DQN suffers from instability and function approximation errors. Experience replay enhances sample efficiency, while target networks improve value stability. However, their combination provides the best learning efficiency, making it the most effective strategy for training stable and high-performing DQN agents.

4.2. Weaknesses

Despite these improvements, several challenges remain. The experiments were limited to the CartPole environment, which is relatively simple and deterministic. More complex environments with high-dimensional state spaces or sparse rewards may require additional improvements such as dueling architectures, double Q-learning, or prioritized experience replay.

The selected hyperparameters, though optimized for CartPole, may not generalize well across different tasks. While the study systematically tested various hyperparameters, the search space was limited, and alternative optimization methods such as Bayesian hyperparameter tuning could yield better results.

One limitation of experience replay is that it treats all past transitions equally, potentially slowing down adaptation to new strategies. Prioritized experience replay could address this by assigning higher importance to more informative transitions. Similarly, target networks introduce a delay in learning updates, which, while stabilizing, may slow down the adaptation to dynamic environments.

Another potential issue is sample inefficiency. While experience replay helps, deep reinforcement learning remains highly sample-inefficient. The experiments required 1,000,000 training steps per run, which may be impractical for real-world applications where data collection is expensive.

In summary, while this study provides insights into hyperparameter tuning and stabilization techniques for DQN, additional improvements are needed to enhance generalization, sample efficiency, and adaptability to more complex environments.

References

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Lin, L.-J. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8(3-4):293–321, 1992.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. MIT Press, 2 edition, 2018.

van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30, 2016a.

van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pp. 2094–2100, 2016b.

Watkins, C. J. and Dayan, P. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

Zhang, S. and Sutton, R. S. A deeper look at experience replay. *arXiv preprint arXiv:1712.01275*, 2017.