Reinforcement Learning on Taxi-v3: A Study of Tabular Q-Learning and Deep Q-Networks

Abstract

This report presents a comprehensive implementation and comparative analysis of two fundamental reinforcement learning algorithms applied to the Taxi-v3 environment from Gymnasium. We implement and evaluate both classical Tabular Q-Learning and Deep Q-Networks (DQN), investigating their performance, convergence properties, and stability characteristics. The Taxi-v3 environment presents a discrete navigation and planning problem with 500 states and 6 actions, providing an ideal testbed for comparing tabular and function approximation methods. Our experimental results demonstrate that both algorithms successfully learn effective policies, with Tabular Q-Learning achieving a mean test reward of 8.48 (±2.47) and DQN achieving 7.95 (±2.60) across multiple runs. The implementation incorporates modern RL techniques including epsilon-greedy exploration with decay schedules, experience replay, target networks, and action masking for improved sample efficiency.

Introduction

Reinforcement Learning (RL) has emerged as a powerful paradigm for training agents to make sequential decisions in complex environments. Unlike supervised learning, RL agents learn through interaction with an environment, receiving rewards or penalties based on their actions. This trial-and-error learning process enables agents to discover optimal policies without requiring explicit labeled training data.

The Taxi-v3 environment, part of the Gymnasium suite, represents a classic problem in discrete reinforcement learning. In this environment, a taxi agent must navigate a 5×5 grid world to pick up passengers from one of four designated locations (Red, Green, Yellow, Blue) and drop them off at their desired destinations. The agent receives positive rewards for successful passenger delivery and negative rewards for incorrect pickup/drop-off attempts and for each time step without completing the task.

This project implements and compares two fundamental approaches to solving this problem:

- **Tabular Q-Learning**: A classical value-based RL algorithm that maintains an explicit table of state-action values
- **Deep Q-Networks (DQN)**: A modern deep learning approach that uses neural networks to approximate the Q-value function

The comparison between these methods provides insights into the trade-offs between tabular and function approximation approaches in discrete state spaces.

Problem Definition

The Taxi-v3 Environment

The Taxi-v3 environment presents a navigation and planning challenge with the following characteristics:

State Space: The environment has 500 discrete states, encoded as a single integer representing:

- 25 possible taxi positions in a 5×5 grid
- 5 possible passenger locations (Red, Green, Yellow, Blue, or inside the taxi)
- 4 possible destination locations (Red, Green, Yellow, Blue)

State encoding formula:

((taxi_row × 5 + taxi_col) × 5 + passenger_location) × 4 + destination

Action Space: The agent can choose from 6 discrete actions at each time step:

0. Move South (down)
1. Move North (up)
2. Move East (right)
3. Move West (left)
4. Pick up passenger
5. Drop off passenger

Reward Structure:

- +20 points for successful passenger delivery
- -10 points for illegal pickup/drop-off attempts
- -1 point for each time step (movement penalty)

This reward structure encourages the agent to complete tasks quickly while avoiding incorrect actions.

Episode Termination: An episode ends when the passenger is successfully dropped off at the correct destination, or after a maximum number of steps.

Solution Approach

Tabular Q-Learning

Tabular Q-Learning is a model-free, value-based reinforcement learning algorithm that learns the optimal action-value function through iterative updates. The algorithm maintains a Q-table where each entry $Q(s, a)$ represents the expected cumulative reward for taking action a in state s and following the optimal policy thereafter.

Algorithm: The Q-value update rule follows the Bellman equation:

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

where:

- α is the learning rate
- γ is the discount factor (0.99)
- r is the immediate reward
- s' is the next state
- $\max_{a'} Q(s', a')$ is the maximum Q-value in the next state

Exploration Strategy: The agent uses an epsilon-greedy policy with exponential decay:

- Initial ϵ = 1.0 (pure exploration)
- Minimum ϵ = 0.05 (5% exploration maintained)
- Decay rate = 0.99988 per episode

This schedule ensures extensive early exploration while gradually shifting toward exploitation of learned knowledge.

Action Masking Enhancement: We use action masks provided by the Taxi-v3 environment. The *action_mask* field in the environment's info dictionary indicates which actions actually change the state (e.g., moving into a wall is masked out). By restricting both random exploration and greedy exploitation to valid actions only, we significantly accelerate learning and avoid wasted experience on no-op transitions.

Deep Q-Network (DQN)

Deep Q-Networks extend Q-learning to high-dimensional state spaces by using neural networks as function approximators.

Network Architecture: Our Q-network is a multi-layer perceptron (MLP) with the following structure:

Layer	Input Size	Output Size
Input (One-Hot Encoding)	500	500
Hidden Layer 1 (ReLU)	500	128
Hidden Layer 2 (ReLU)	128	128
Output Layer (Linear)	128	6

Table 1: DQN neural network architecture

The network takes a one-hot encoded state vector (500 dimensions) as input and outputs Q-values for all 6 actions.

Key DQN Techniques:

- **Experience Replay**: Transitions $(s, a, r, s'$,done$)$ are stored in a replay buffer (capacity: 50,000). During training, random mini-batches (size: 64) are sampled to break temporal correlations and improve sample efficiency.
- **Target Network**: A separate target network with frozen weights is used to compute TD (Temporal Difference) targets, stabilizing training by reducing the correlation between current and target Q-values. The target network is synchronized with the main network every 500 training steps.
- **Linear Epsilon Decay**: Unlike the exponential decay in tabular Q-learning, DQN uses linear decay over 50,000 steps from $\epsilon = 1.0$ to $\epsilon = 0.05$.
- **Gradient Clipping**: Gradients are clipped to a maximum norm of 1.0 to prevent exploding gradients during training.
- **Smooth L1 Loss**: Also known as Huber loss, this loss function is less sensitive to outliers than mean squared error, improving training stability.

Training Pipeline:

- **Warm-up phase**: Collect 1,000 transitions before beginning training
- **Interaction**: Agent selects actions using epsilon-greedy policy
- **Storage**: Transitions stored in replay buffer with action masks
- **Training**: Sample mini-batch and perform gradient descent every step
- **Synchronization**: Update target network every 500 steps

Implementation Architecture

Project Structure

File	Purpose
main.py	Orchestrates training and testing workflows
tabular_qlearning.py	Tabular Q-Learning agent implementation
dqn_agent.py	Deep Q-Network agent implementation
plot_utility.py	Visualization and plotting functions

Table 2: Project file organization

Hyperparameters

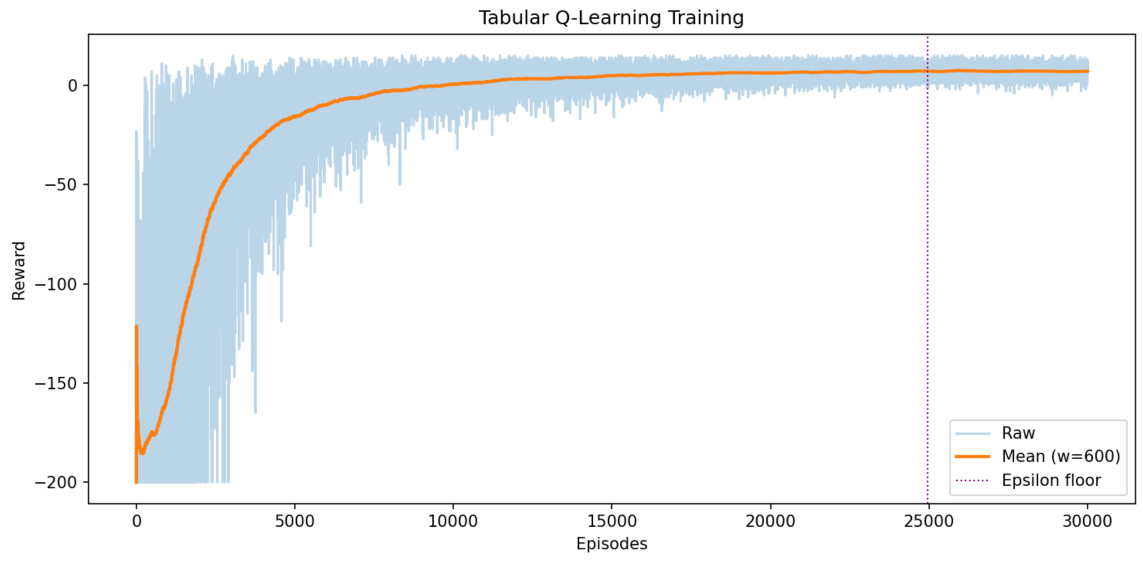
Parameter	Tabular Q-Learning	DQN
Learning Rate (α)	0.7	0.001
Discount Factor (γ)	0.99	0.99
Initial Epsilon (ϵ_0)	1.0	1.0
Minimum Epsilon ($\epsilon_{\min}$)	0.05	0.05
Epsilon Decay	0.99988 (exp)	50,000 steps (linear)
Training Episodes	30,000	10,000
Test Episodes	100	100

Table 3: Hyperparameter configuration for both algorithms.

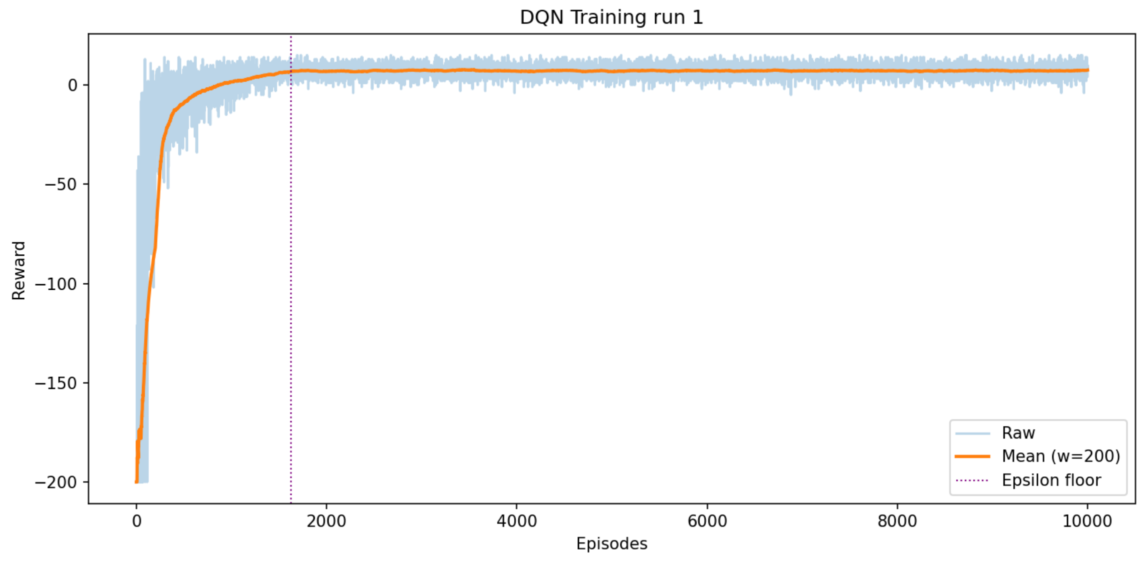
Although learning-rate decay parameters are defined in the code, LR decay is currently disabled (DECAY_LR = False), so α remains fixed at 0.7 and 0.001 during training.

Experimental Results

Training Performance



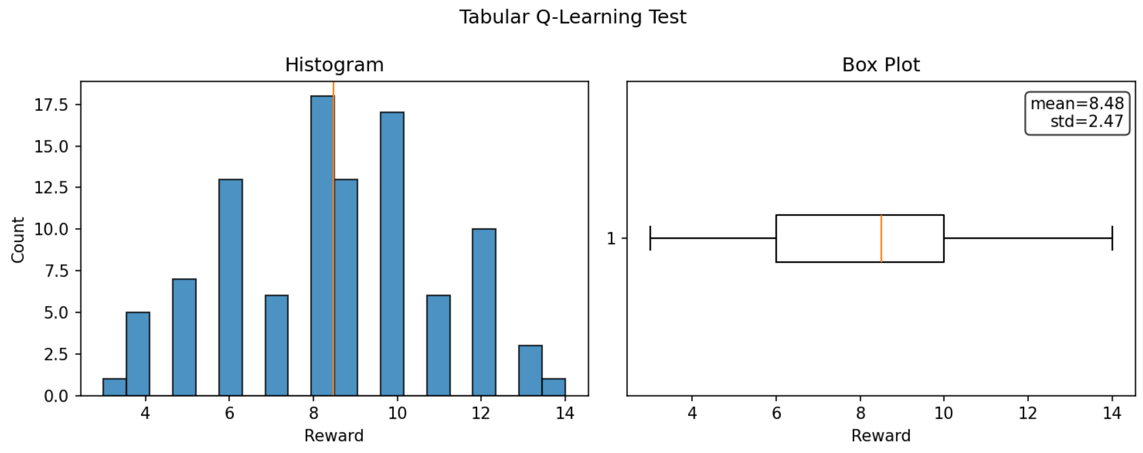
Tabular Q-Learning: The training curve shows rapid initial learning with the mean reward converging from approximately -200 to very good performance within the first 10,000 episodes. The epsilon floor (where ϵ reaches its minimum value) is reached around episode 25,000, after which the agent maintains 95% exploitation. The smoothed curve (window size: 600) demonstrates stable convergence with minimal variance in the later training stages.



Deep Q-Network: DQN exhibits faster initial convergence compared to tabular Q-learning, reaching near-optimal performance within 2,000-3,000 episodes across all three runs. This acceleration is attributed to the combination of experience replay (enabling multiple updates per transition) and the representational power of the neural network. The epsilon floor is reached around episode 2,000 for all runs.

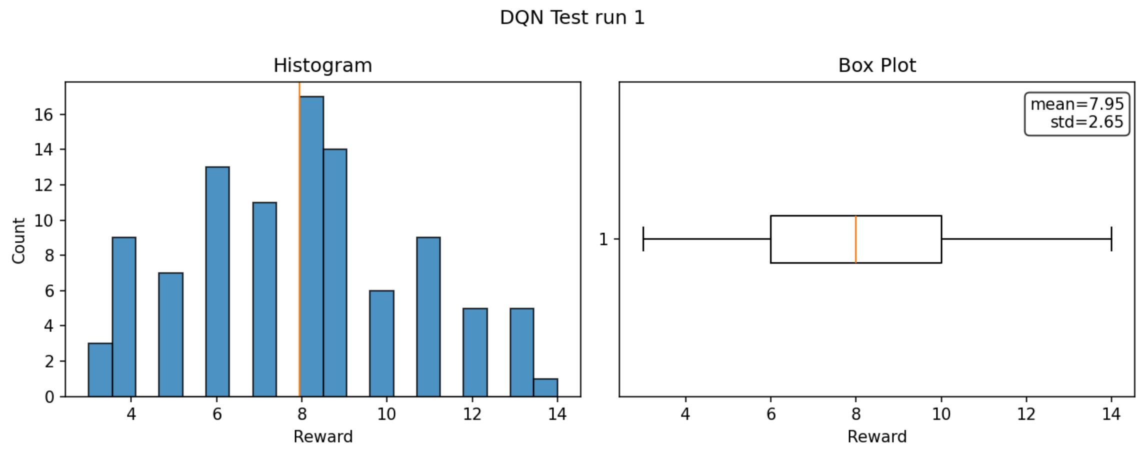
Test Performance

We evaluate both agents using greedy policies (epsilon = 0) over 100 independent test episodes.



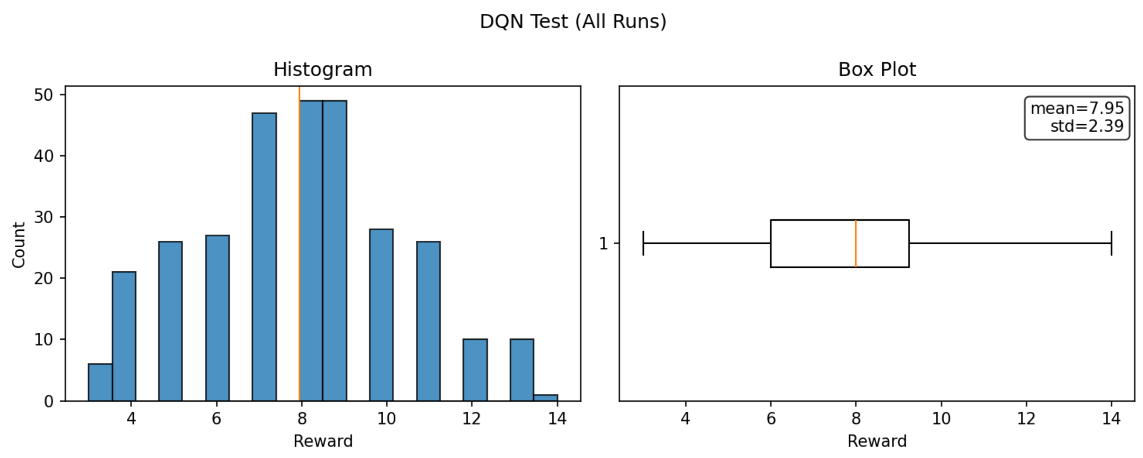
Tabular Q-Learning Test Results:

- Mean reward: 8.48
- Standard deviation: 2.47
- Reward range: Approximately 3 to 14
- Distribution: Right-skewed with mode around 8-10



DQN Test Results (Single Run):

- Mean reward: 7.95
- Standard deviation: 2.65
- Reward range: Approximately 3 to 14
- Distribution: Centered around 8-9



DQN Test Results (All 3 Runs Combined, 300 episodes):

- Mean reward: 7.95
- Standard deviation: 2.39
- Reward range: Approximately 2 to 15
- Distribution: Centered around 8-9

DQN Multi-Run Stability:

Across three independent training runs, DQN demonstrates consistent performance:

- Overall mean: 7.95
- Inter-run standard deviation: 0.31

The low inter-run standard deviation (0.31) indicates that DQN training is stable and reproducible despite the stochastic nature of neural network optimization.

Comparative Analysis

Metric	Tabular Q-Learning	DQN (3 runs)
Test Mean Reward	8.48	7.95
Test Std Deviation	2.47	2.60
Training Episodes to Convergence	~ 5,000	~ 2,000
Sample Efficiency	Lower	Higher
Memory Requirements	3,000 floats	~ 81,000
Inter-Run Stability	N/A	0.31 std

Table 4: Performance comparison between algorithms

Key Observations:

1. **Final Performance**: Tabular Q-Learning obtains a slightly higher mean test reward than DQN (8.48 vs 7.95.
2. **Convergence Speed**: DQN converges approximately 2.5× faster in terms of episodes (2,000 vs 5,000, although each episode involves significantly more computation due to replay updates) demonstrating the benefits of experience replay and neural function approximation.
3. **Variance**: Both methods exhibit similar test reward variance (2.47 vs 2.60), indicating comparable policy stability.
4. **Reproducibility**: DQN shows excellent reproducibility across runs (inter-run std = 0.31), suggesting that the algorithm is robust to initialization and sampling randomness.
5. **Computational Efficiency**: Tabular Q-Learning is more computationally efficient per episode due to simple table lookups, while DQN requires forward/backward passes through a neural network.

Optimal Performance Benchmark

We use the default Gymnasium Taxi-v3 configuration (max_episode_steps = 200).

In this setting, the registered reward threshold is 8.0, with reported optimal performance around 8.46.

Therefore, our Tabular Q-Learning result (8.48) exceeds the benchmark threshold, while DQN (7.95) remains slightly below it.

This indicates that both methods learn strong policies, with tabular performance meeting the environment’s benchmark criterion.

Discussion

Algorithm Comparison

The experimental results reveal nuanced trade-offs between tabular and deep reinforcement learning approaches in the Taxi-v3 environment.

When Tabular Q-Learning Excels:

- **Small State Spaces**: With only 500 states, the tabular approach is perfectly suited, avoiding unnecessary approximation errors
- **Simplicity**: No hyperparameters related to neural network architecture, no concerns about overfitting or gradient instability
- **Interpretability**: The Q-table can be directly inspected to understand learned values

- **Computational Efficiency**: Table lookups are orders of magnitude faster than neural network inference

When DQN Shows Advantages:

- **Sample Efficiency**: Experience replay enables multiple gradient updates per environment transition, accelerating convergence
- **Scalability Potential**: While overkill for Taxi-v3, the same DQN architecture can scale to environments with millions of states (e.g., more sophisticated games)
- **Generalization**: Neural networks can potentially generalize to unseen states, though this benefit is minimal in fully explorable discrete environments

Action Masking Impact

The implementation of action masking based on environment feedback represents a significant enhancement to both algorithms. By restricting action selection to only valid actions:

- Reduced wasted experience on no-op transitions (e.g., attempting to move into walls)
- Accelerated convergence by focusing learning on meaningful state transitions
- Improved final policy quality by avoiding meaningless actions during both training and testing

This technique is particularly valuable in environments with structured constraints and should be considered a best practice when such information is available.

Conclusion

This project successfully implements and compares two fundamental reinforcement learning paradigms on the Taxi-v3 environment. Both Tabular Q-Learning and Deep Q-Networks demonstrate strong learning capabilities, achieving mean test rewards of 8.48 and 7.95 respectively.

Key findings include:

- DQN converges 2.5× faster than tabular methods due to experience replay
- Tabular Q-Learning achieves marginally higher final performance
- Action masking significantly improves sample efficiency for both algorithms
- DQN exhibits excellent reproducibility across independent training runs (inter-run std = 0.31)

For the Taxi-v3 environment specifically, the tabular approach represents the more appropriate choice due to its simplicity, interpretability, and computational efficiency. However, the DQN implementation provides valuable insights into deep reinforcement learning techniques that are essential for tackling more complex environments with high-dimensional state spaces.

References

You can find here the repository of the project: https://github.com/giuliodalbono/ML-Project