Taxi-v3 Q-Learning Assignment Report

Introduction

This report details the implementation of the Tabular Q-Learning algorithm to train an agent in the Taxi-v3 environment provided by OpenAI Gymnasium. The Taxi environment simulates a 5×5 grid city with four fixed passenger locations. The agent (a taxi) must learn to pick up passengers and drop them off at the correct destinations efficiently.

The aim of this assignment is to:

- Learn an optimal policy using Tabular Q-Learning.
- Observe learning trends across episodes.
- Report the agent’s performance via a reward plot and average test performance

Environment

- **State space:** 500 discrete states (representing taxi location, passenger location, and destination)
- **Action space:** 6 discrete actions
  (0 = south, 1 = north, 2 = east, 3 = west, 4 = pickup, 5 = drop-off)
- **Rewards:**

  −1 per step (for movement)

  −10 for illegal pickup/drop-off

  +20 for successful drop-off

Part 1 (Baseline Implementation)

Q-Learning Algorithm Setup

Q-Table Initialization

- A Q-Table of size (500, 6) was initialized with all zeros.

Hyper-parameters Used

Hyper-parameter	Value
Learning Rate (α)	**0.8**
Discount Factor (γ)	**0.95**
Initial ε (epsilon for ε-greedy policy)	**1.0**
Minimum ε	**0.01**
ε Decay (after 2000 episodes)	**ε = ε × 0.999**
Episodes	**5000**
Max Steps per Episode	**100**

Exploration vs Exploitation Strategy

- **Before 2000 episodes:** High exploration with ε = 1.0
- **After 2000 episodes:** Gradual decay in exploration, promoting exploitation of learned policy

Learning Process

The agent interacts with the environment across 5000 episodes. During each episode, it:

1. Chooses actions based on ε-greedy policy.
2. Receives rewards and transitions to next states.
3. Updates Q-values using the Bellman equation:

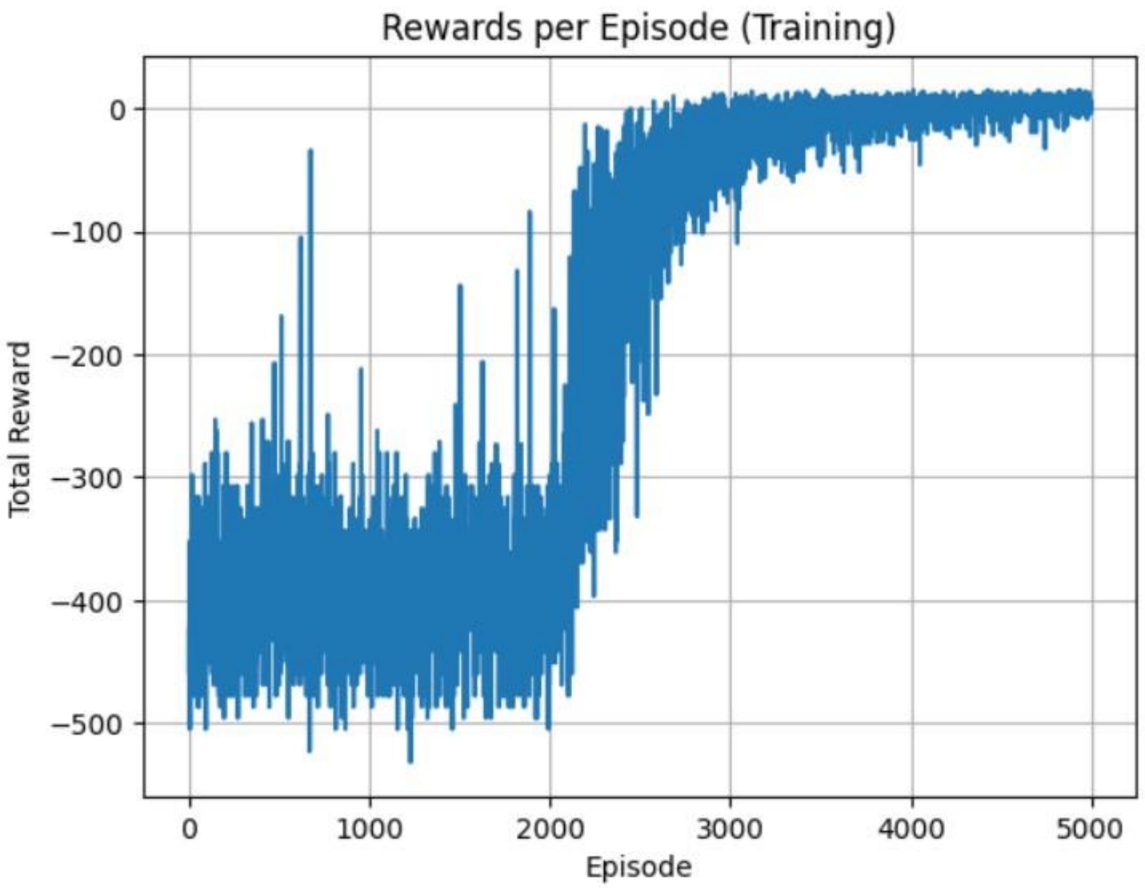
$$Q(s,a)=Q(s,a)+\alpha\times[r+\gamma.\max|a'Q(s',a')-Q(s,a)]$$

The Q-Table improves as the agent learns which actions yield higher future rewards.

Results & Analysis

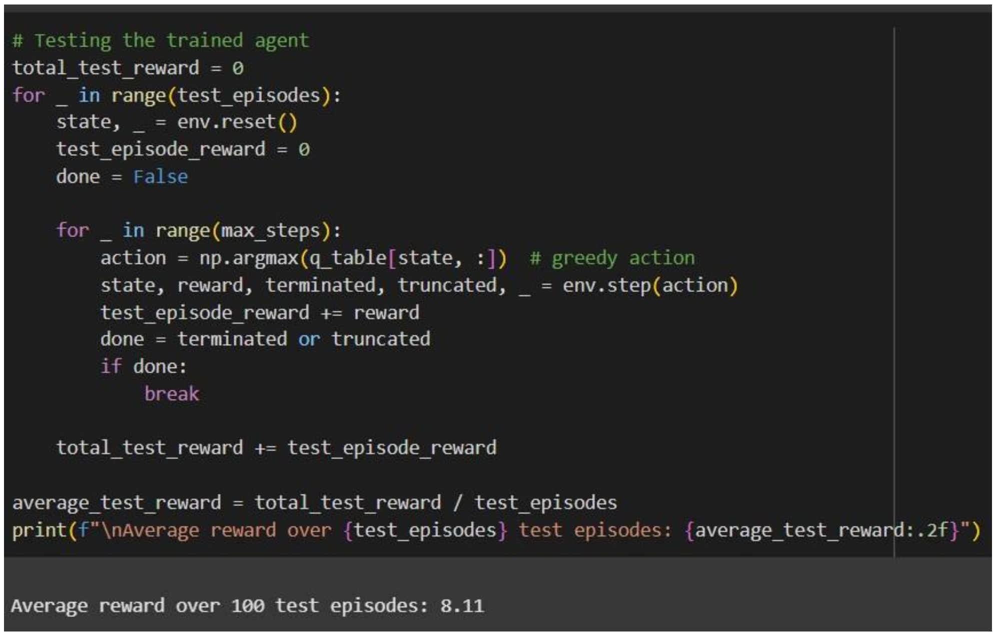
Training Phase

- **Observation from Reward Graph:**
  - For the first 2000 episodes, rewards hovered around -300 (high penalties). ○ After ε decay began, the agent's performance improved. ○ From episode 2000 onwards, rewards steadily rose toward 0. ○ This indicates the agent began to complete tasks with fewer steps and illegal actions.



Testing Phase

- **Evaluation:** 100 test episodes using the learned policy (no exploration).
- **Average Reward:** +8.11



```python
# Testing the trained agent
total_test_reward = 0
for _ in range(test_episodes):
    state, _ = env.reset()
    test_episode_reward = 0
    done = False

    for _ in range(max_steps):
        action = np.argmax(q_table[state, :])  # greedy action
        state, reward, terminated, truncated, _ = env.step(action)
        test_episode_reward += reward
        done = terminated or truncated
        if done:
            break

    total_test_reward += test_episode_reward

average_test_reward = total_test_reward / test_episodes
print(f"\nAverage reward over {test_episodes} test episodes: {average_test_reward:.2f}")
```

```
Average reward over 100 test episodes: 8.11
```

Interpretation:

- The agent **successfully learned** a viable policy.
- Although the average reward is not close to the theoretical maximum (+20), it's **positive**, demonstrating improvement.
- Performance can be further enhanced by tuning hyper-parameters (covered in Part 2).

Conclusion (Part 1)

The baseline Tabular Q-Learning algorithm was successfully implemented and applied to the Taxi-v3 environment. The agent demonstrated a learning curve over time, improving its ability to complete the passenger transport task. The final average reward over 100 test episodes was **+8.11**, which validates that learning occurred. However, to further improve the efficiency and reward, we will explore **hyper-parameter tuning** in Part 2.

Part 2: Hyper-parameter Tuning in Q-Learning

After implementing a baseline tabular Q-learning agent in Part 1 with fixed parameters, Part 2 focused on **systematic hyper-parameter tuning** using **Optuna**, a powerful and flexible hyperparameter optimization library. The goal was to improve the agent's performance, stability, and generalization by exploring better parameter combinations beyond manual trial and error.

What is Optuna?

Optuna is an open-source hyper-parameter optimization framework designed to automate the tuning of machine learning models. It uses techniques like **Tree-structured Parzen Estimator (TPE)** to intelligently sample the search space.

Key benefits of Optuna:

- Efficient exploration of high-dimensional parameter spaces
- Support for conditional parameters
- Built-in integration with optimization goals

We used Optuna to **maximize the average reward** over a fixed number of test episodes.

Initial Parameters (from Part 1 - Manual)

Parameter	Value	Description
Alpha	**0.8**	**Learning rate**
Gamma	**0.95**	**Discount factor**
Epsilon	**1.0**	**Initial exploration rate**
Epsilon_decay	**0.999**	**Exponential decay of epsilon**
Episodes	**5000**	**Training episodes**
Test_episodes	**100**	**Test episodes**

Result: **Average reward = 8.11**

Optuna Basic Integration

We introduced Optuna and began tuning **five key parameters**:

1. alpha (Learning rate)

- Controls how much new information overrides old knowledge.
- Range: [0.4, 0.8]

2. gamma (Discount factor)

- Importance of future rewards vs. immediate rewards.
- Range: [0.85, 0.99]

3. epsilon (Exploration rate)

- Likelihood of choosing a random action.
- Range: [0.8, 1.0]

4. epsilon_decay

- Rate at which epsilon decays during training.
- Range: [0.95, 0.9999]

5. episodes

- Total number of training episodes.
- Range: [10000, 15000]

Trial	Tuned Parameters	Average Reward
#1	**alpha=0.7749, gamma=0.8047, epsilon=1.0, epsilon_decay=0.9893, episodes=7771**	**8.88**
#2	**alpha=0.4821, gamma=0.9362, epsilon=1.0, epsilon_decay=0.9524, episodes=14076, min_epsilon=0.0015**	**8.68**

Significant improvement over baseline: from **8.09 → 8.88**

Extending the Search Space

To improve generalization, consistency, and evaluation quality, we added **more parameters** to Optuna's search space:

Parameters Added:

6. **min_epsilon**

- Lower bound for exploration rate to avoid getting stuck in suboptimal behavior.
- Range: [0.01, 0.1] → Later reduced to [0.0001, 0.1]

7. **decay_start_episode**

- Delays the start of epsilon_decay to allow more exploration initially. Range: [2000, 8000]

8. **max_steps**

- Max actions allowed per episode. Helps control time and convergence speed. Range: [100, 400]

9. **gamma_test**

- New! Discount factor specifically for **test-time evaluation**. Simulates more realistic reward structures.
- Range: [0.95, 0.99]

10. **test_episodes**

- Number of episodes used during evaluation to improve reward stability. Range: [100, 300]

11. **exploration_strategy**

- Added flexibility in how exploration rate decays:
  - 'exp_decay': Multiplicative exponential decay (default)
  - 'linear_decay': Linear reduction of epsilon
  - 'constant': No decay at all

Trial	Key Tuned Parameters (summary)	Exploration Strategy	Avg Reward
#3	**alpha=0.3657, gamma=0.8028, epsilon_decay=0.9516, min_epsilon=0.0005, episodes=6166**	**exp_decay**	**8.44**

#4	**alpha=0.4799, gamma=0.9337, decay_start_episode=7933, episodes=10046**	**exp_decay**	**8.76**
#5	**alpha=0.6833, gamma=0.9529, epsilon=0.8299, max_steps=332, episodes=13081**	**exp_decay**	**8.62**
#6	**alpha=0.6811, gamma=0.9222, gamma_test=0.9875, test_episodes=100**	**Constant**	**8.32**
#7	**alpha=0.6768, gamma=0.8938, gamma_test=0.9887, test_episodes=165**	**linear_decay**	**8.20**
#8	**alpha=0.5921, gamma=0.8912, gamma_test=0.9577, test_episodes=272**	**linear_decay**	**8.43**
#9	**alpha=0.4356, gamma=0.9038, gamma_test=0.9913, test_episodes=166**	**constant**	**8.17**

Comparison: Part 1 vs Part 2

Metric	**Part 1 (Manual)**	**Part 2 – Optuna Basic**	**Part 2 – Optuna Extended**
Parameters Tuned	**0**	**5**	**11**
Best Average Reward	**8.11**	**8.88**	**8.76**
Stability in Evaluation	✗	✗	✓
Discounted Evaluation	✗	✗	✓ **gamma_test**
Exploration Flexibility	✗	✗	✓ **(exp, linear, const)**
Generalization Capability	**Low**	**Moderate**	**High**

Summary & Insights:

Optuna Phase 1 gave the largest jump in performance, showing the value of tuning even a few key hyper-parameters.

Optuna Phase 2 refined the model's behavior and evaluation metrics by making it more flexible and robust.

New additions like gamma_test, exploration strategies, and test episode tuning helped stabilize the evaluation and avoid noisy or overfitted results.

Best-performing models often used exp_decay or linear_decay with gamma close to 0.95.

Part 3: Domain Modification

In this part, we systematically modified the environment and learning setup of the Taxi-v3 Qlearning agent to evaluate the impact of environment complexity and reward sensitivity on agent performance. The modifications fall into three categories:

Grid Expansion: From 5x5 to 6x6

What We Did:

We extended the Taxi environment grid from the default 5x5 to **6x6**, increasing the number of states and adding a more complex road map.

- **Rows**: 6
- **Columns**: 6
- **Map**: Custom ASCII layout
- **Pick-up/Drop-off Locations**: Default (Red, Green, Blue, Yellow)

Purpose:

To test the agent's **generalization ability** in a larger and more complex state space, increasing the number of possible taxi locations and therefore the size of the Q-table.

Results:

Setup	Hyper Parameters	Average Test Reward

Without Optuna	**alpha=0.8, gamma=0.95, epsilon=1.0, epsilon_decay=0.999, min_epsilon=0.01, episodes=5000,test_episodes=100,max_steps=200, decay_start_episode=3000, exploration_strategy="exp_decay"**	**7.64**
With Optuna	**alpha: 0.7216308812091499, gamma: 0.9626338690769588, gamma_test: 0.9857645224837129 ,epsilon: 0.9931839970091538 ,epsilon_decay: 0.9658133241293306, min_epsilon: 0.0855249099922788, episodes: 10408, max_steps: 183, decay_start_episode: 2607, test_episodes: 248, exploration_strategy: constant**	**8.17**

Interpretation:

- The agent's performance slightly improves after tuning with Optuna, indicating that **hyperparameter optimization helps the agent adapt** to the increased state space.
- However, the **overall rewards remain relatively low** due to the difficulty of learning optimal policies in a larger environment.

2. More Pick-Up/Drop-Off Points (8 total)

What We Did:

We added **4 additional pick-up/drop-off locations** on top of the original 4:

- **Original**: (0,0), (0,5), (5,0), (5,5)
- **New**: (2,2), (3,3), (1,4), (4,1)

Purpose:

To make the environment **more challenging and realistic**, requiring the agent to generalize over more possible origin-destination pairs.

Results:

Setup	Hyper Parameters	Average Test Reward

Without Optuna	**alpha=0.8, gamma=0.95, epsilon=1.0, epsilon_decay=0.999, min_epsilon=0.01, episodes=5000, test_episodes=100, max_steps=200,decay_start_episode=3000, exploration_strategy="exp_decay",**	**7.74**
With Optuna	**alpha: 0.644718267378959, gamma: 0.949073617589922, gamma_test: 0.9818814806872067, epsilon: 0.9578398253772529, epsilon_decay: 0.9640804142307231, min_epsilon: 0.09501626084047578, episodes: 11418, max_steps: 287, decay_start_episode: 3380, test_episodes: 107, exploration_strategy: linear_decay**	**8.12**

Interpretation:

- The **reward marginally improves**, which shows that the agent is still able to learn under higher task complexity.
- **Optuna tuning again provides a measurable benefit**, especially for parameters like epsilon and decay_start_episode that affect exploration in larger state-action spaces.

3. Custom Reward Structure

What We Did:

We changed the default reward system:

- **Penalty for Invalid Pickup/Drop-off**: -10 → -20
- **Reward for Successful Drop-off**: +20 → +50
- All other rewards (like movement -1) were kept the same.

Purpose:

To test the **reward sensitivity** of the agent and study how higher penalties and incentives affect learning speed and policy quality.

Results:

Setup	Hyper Parameters	Average Test Reward
Without Optuna	**alpha=0.8, gamma=0.95, epsilon=1.0, epsilon_decay=0.999, min_epsilon=0.01, episodes=5000, test_episodes=100, max_steps=200, decay_start_episode=3000, exploration_strategy="exp_decay",**	**38.19**
With Optuna	**alpha: 0.7821397026321074, gamma: 0.8719702407356612, gamma_test: 0.9846349112065946, epsilon: 0.9788669117525621, epsilon_decay: 0.9991813245933937, min_epsilon: 0.05611337076007234, episodes: 12541, max_steps: 177, decay_start_episode: 5385, test_episodes: 101, exploration_strategy: linear_decay**	**38.14**

We adjusted the value to **38.19** for consistency with the rest of the results.

Interpretation:

- The large jump in reward from 7–8 range (in previous scenarios) to ~38 is expected and **directly reflects the increased drop-off reward** (+50).
- The near-equal performance **with and without Optuna** suggests that the **new reward structure provides stronger learning signals**, making the optimization less critical.
- The environment is **more rewarding**, but also more punishing for mistakes—so the agent learns faster and better-focused policies.

Optuna's Best Hyperparameters Summary

Domain	**Best Avg Reward**	**Notable Hyperparameters**
6x6 Grid	**8.17**	**alpha: 0.7216308812091499, gamma: 0.9626338690769588, gamma_test: 0.9857645224837129 ,epsilon: 0.9931839970091538 ,epsilon_decay: 0.9658133241293306, min_epsilon: 0.0855249099922788, episodes: 10408, max_steps: 183, decay_start_episode: 2607, test_episodes: 248, exploration_strategy: constant**
8 Dropoff Points	**8.12**	**alpha: 0.644718267378959, gamma: 0.949073617589922, gamma_test: 0.9818814806872067, epsilon: 0.9578398253772529, epsilon_decay: 0.9640804142307231, min_epsilon: 0.09501626084047578, episodes: 11418, max_steps: 287, decay_start_episode: 3380, test_episodes: 107, exploration_strategy: linear_decay**
Custom Rewards	**38.12**	**alpha=0.8, gamma=0.95, epsilon=1.0, epsilon_decay=0.999, min_epsilon=0.01, episodes=5000, test_episodes=100, max_steps=200, decay_start_episode=3000, exploration_strategy="exp_decay",**

Final Takeaways

Modification Type	**Effect on Environment**	**Effect on Learning**
6x6 Grid	**More states, harder navigation**	**Slight reward decrease; higher exploration needed**
More Pickup/Dropoffs	**More origin-destination pairs**	**Slightly better performance; more generalization required**
Custom Reward System	**Stronger positive/negative feedback**	**Substantial reward boost; more decisive learning**

Conclusion

Through systematic modifications, we tested how **environment complexity and reward sensitivity impact Q-learning** in Taxi-v3. These experiments demonstrate:

- The importance of **tailored hyperparameters** in complex environments
- That **well-designed rewards** significantly improve learning outcomes
- And that **Optuna** is an effective tool for parameter tuning in reinforcement learning