AUTONOMOUS TAXI AGENT: A REINFORCEMENT LEARNING APPROACH

GHOSH Prabal, ILLAHIBUCCUS SONA Ishfaaq, HAMMED Habeeb Olawale

AGENDA

- Introduction
- Problem Statement
- Experimental Setup
- Overview of Algorithms
- Results and Observations
- Comparison Table
- Key Takeaways
- Conclusion

INTRODUCTION

Reinforcement Learning (RL): A machine learning paradigm where an agent learns by interacting with an environment.

Objective: Train an autonomous taxi agent to **pick up and drop off passengers efficiently.**

Algorithms Compared:

Q-Learning (Off-policy, model-free)	**SARSA** (On-policy, model-free)	**Deep Q-Network (DQN)** (Neural network-based Q-learning)	**Value Iteration** (Model-based, dynamic programming)

PROBLEM STATEMENT

- **Environment**: OpenAI Gym **Taxi-v3**
- **The Taxi-v3 environment is a grid-based game where a taxi agent must:**

  - Pick up a passenger from one of the predefined locations.
  - Navigate to the passenger's destination while avoiding penalties.

- **State Space**
  - The environment has 500 discrete states.
  - **Each state represents a combination of:**
    - The taxi's location (5×5 grid = 25 positions).
    - The passenger's location (one of 4 fixed locations or in the taxi).
    - The passenger's destination (one of 4 locations).
  - The state is a single integer representing these factors.

PROBLEM STATEMENT

- **Action Space**
  - There are **6 discrete actions**:
    - Move South
    - Move North
    - Move East
    - Move West
    - Pickup Passenger
    - Drop off Passenger
- **Rewards**
  - **+20** for successfully dropping off a passenger.
  - **-10** for trying to pick up/drop off incorrectly.
  - **-1** for each movement (to encourage efficiency).

EXPERIMENTAL SETUP

- **Common Parameters:**
  - Number of episodes, step limits, learning rate, discount factor, exploration rate.
- **Performance Metrics:**
  - Reward per episode.
  - Q-table or neural network evaluation.
  - Stability vs. speed of convergence.

ALGORITHMS OVERVIEW

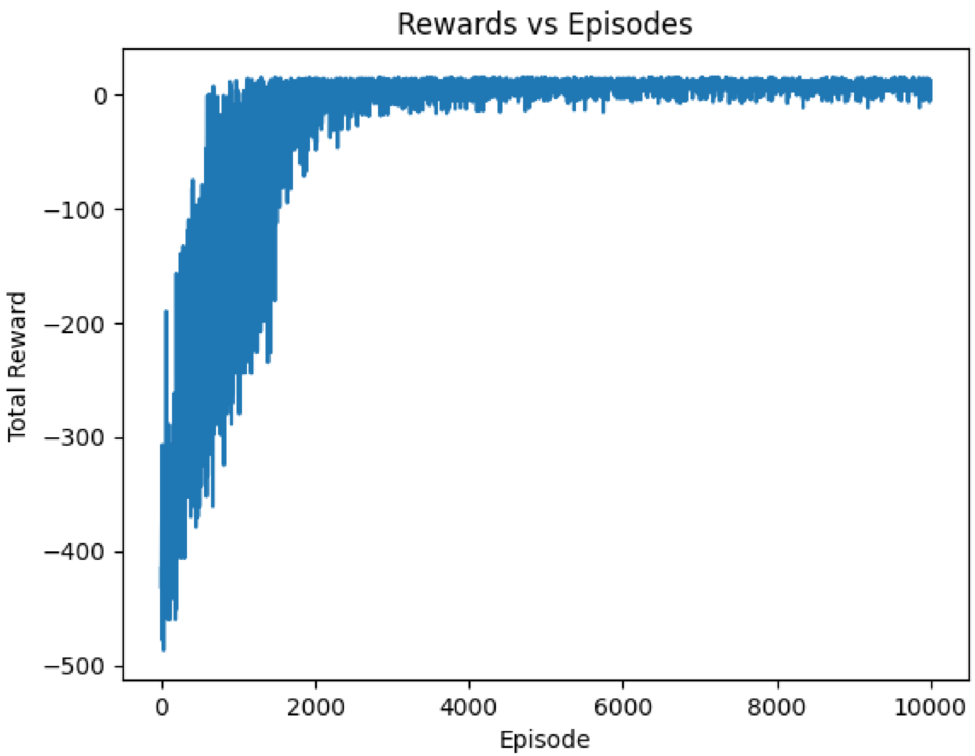
Q-Learning

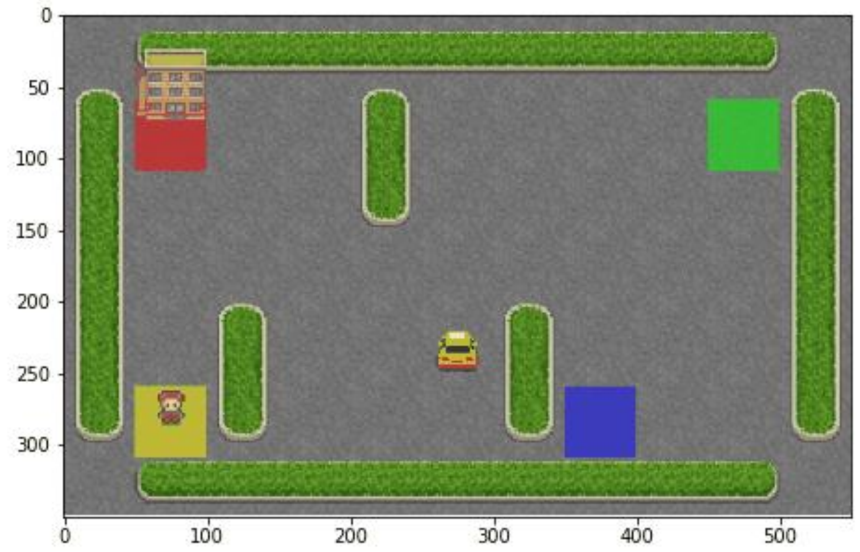
- **Off-policy:** Updates Q-values based on max estimated future reward.
- **Greedy approach:** Faster convergence but less stable.

$$Q^{new}(S_t, A_t) \leftarrow (1 - \underbrace{\alpha}_{\text{learning rate}}) \cdot \underbrace{Q(S_t, A_t)}_{\text{current value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \underbrace{\left(\underbrace{R_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(S_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}_{\text{new value (temporal difference target)}}$$

```
*******Training Finished*******
Q-table shape: (500, 6)
Q-table:
[[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [ 5.45501380e-01  1.72157407e+00 -1.31214161e-01 -1.90504619e+00
   9.62206970e+00 -6.69108357e+00]
 [ 5.40603385e+00  1.32867584e+00 -4.87831254e-01  8.24804817e+00
   1.41188060e+01 -5.46587248e+00]
 ...
 [-1.37793733e+00  2.50584095e+00 -1.33432872e+00 -1.33007409e+00
  -4.83911549e+00 -7.01520742e+00]
 [-3.61720749e+00 -3.55891472e+00 -3.71902320e+00  2.89526122e+00
  -1.09451498e+01 -1.07601504e+01]
 [ 2.83143516e+00  1.14261338e-02  8.86967886e-01  1.78993397e+01
  -2.72881000e+00 -3.04814141e+00]]
Average per thousand episodes
1000 :  -254.47300000000004
2000 :  -40.03799999999996
3000 :  2.221999999999992
4000 :  5.652999999999997
5000 :  6.940999999999965
6000 :  7.334999999999715
7000 :  7.484999999999961
8000 :  7.391999999999965
9000 :  7.300999999999976
10000 :  7.400999999999969
```

Q-LEARNING OUTPUT





ALGORITHMS OVERVIEW

SARSA

- **On-policy**: Uses the actual action taken for updates.
- **More stable** but converges slower than Q-Learning.

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

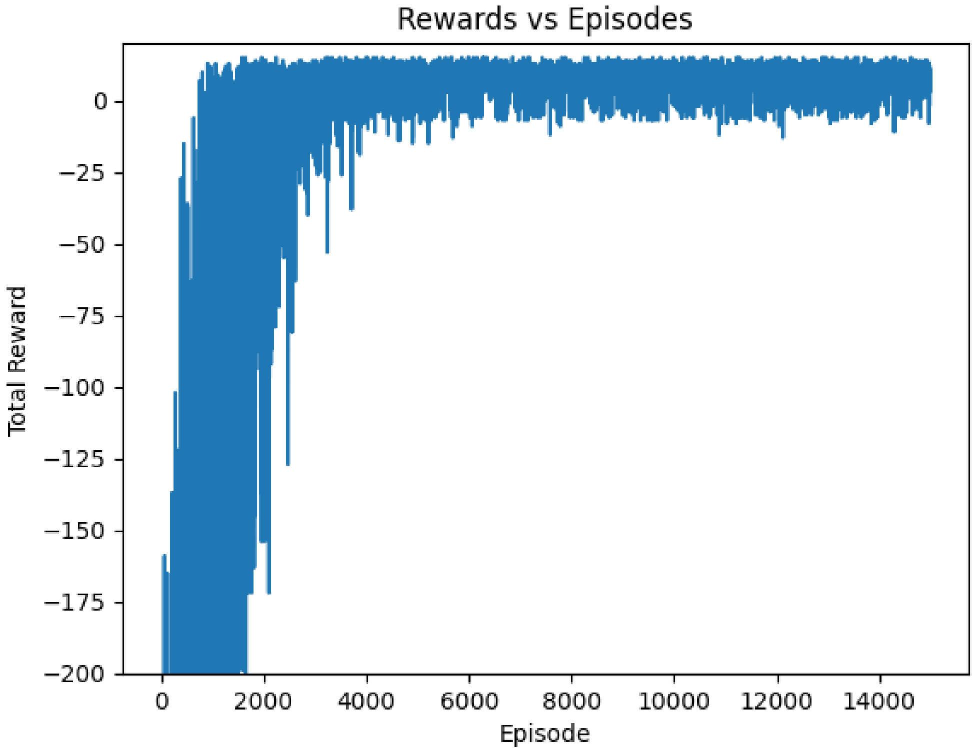
Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

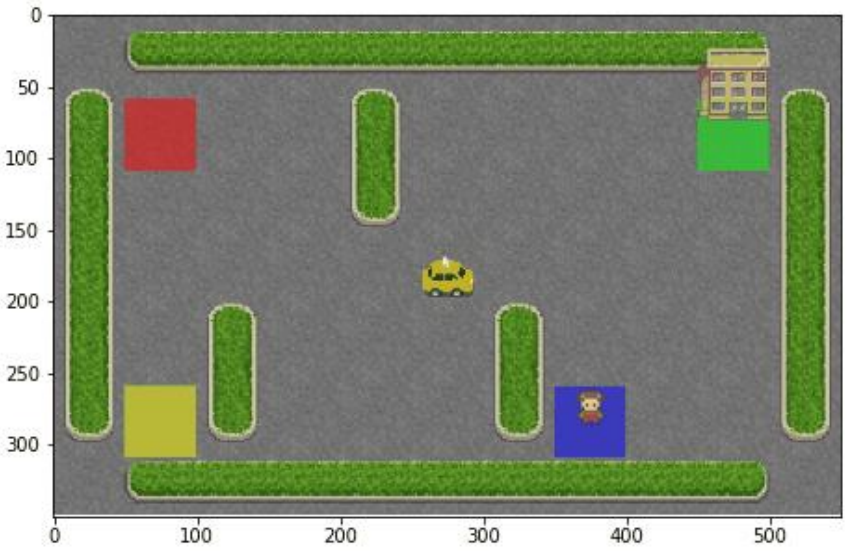
Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

SARSA OUTPUT

```
Q-table shape: (500, 6)
Q-table:
[[  0.           0.           0.           0.           0.
    0.        ]
 [ -5.13381722  -3.58811454  -5.10611238  -6.89387361   8.30154937
  -13.56307411]
 [ -4.63404437  -2.5104213   -7.3552193   -2.58620319  13.68432626
  -12.14939672]
 ...
 [  2.83709581  15.19490515   2.74610547   0.39118843  -7.12668458
   -2.94024222]
 [-10.19108548  -9.45173625  -9.99286757  -9.96893179 -15.19762856
  -15.17785482]
 [  5.04970846   9.51712439   6.9374308   18.79994813  -2.74907938
    2.90143596]]
Average per thousand episodes
1000 :  -254.81399999999996
2000 :  -64.36799999999997
3000 :  -3.896
4000 :  5.291999999999973
5000 :  6.733999999999973
6000 :  7.049999999999966
7000 :  7.13799999999996
8000 :  7.097999999999978
9000 :  7.1419999999999675
10000 :  7.16299999999997
11000 :  7.329999999999963
12000 :  7.09499999999996
13000 :  7.282999999999968
14000 :  7.102999999999965
15000 :  7.193999999999965
```

SARSA OUTPUT



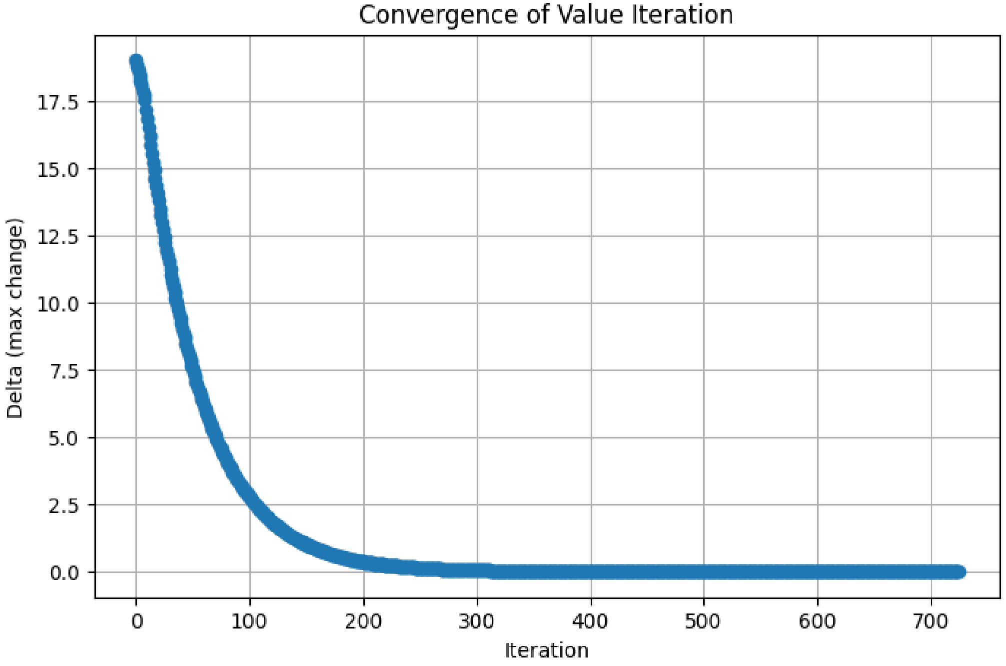


ALGORITHMS OVERVIEW

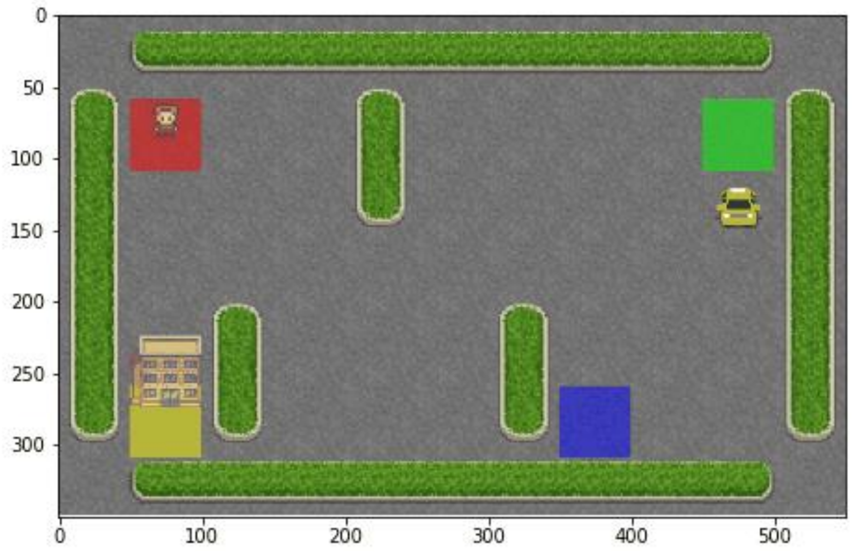
Value Iteration:

- **Model-based**: Uses Dynamic Programming (DP).
- Computes optimal policy iteratively using **Bellman Optimality Equation.**
- **Fast and stable** but computationally expensive in large state spaces.

```
Value Iteration converged in 725 iterations.
```



VALUE ITERATION OUTPUT



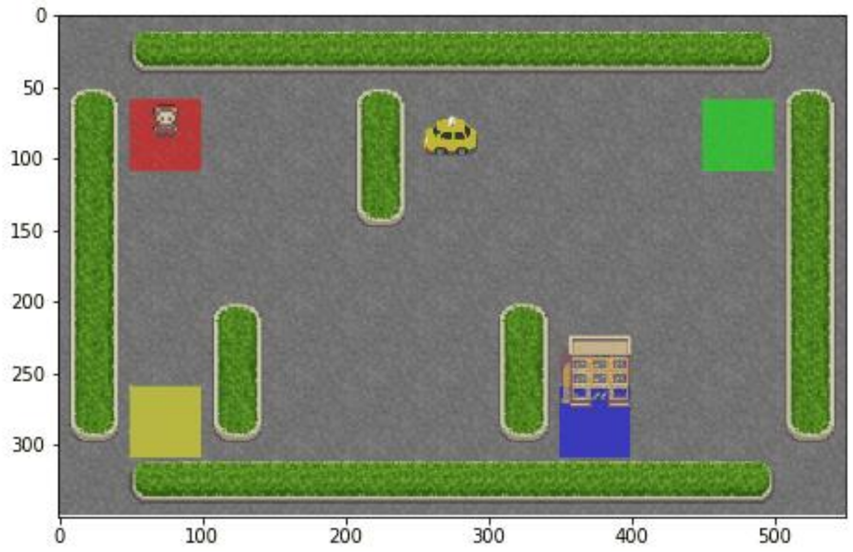
ALGORITHMS OVERVIEW

DQN:

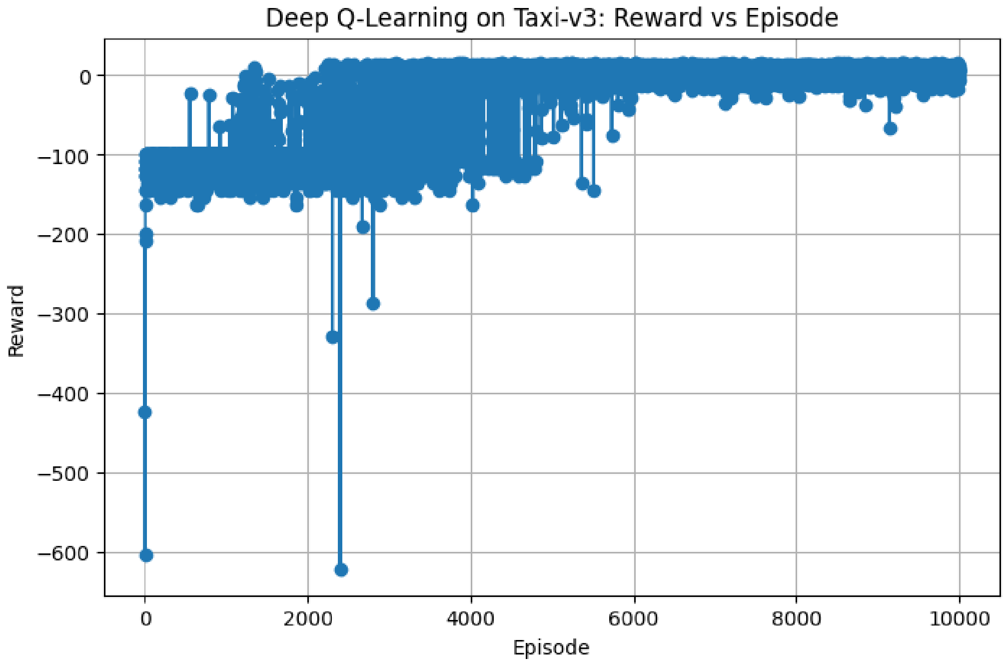
- **Neural network-based Q-learning.**
- Uses **experience replay** and **target networks.**
- Effective in large state spaces but struggles in small environments.

```
Episode 7200: Average Reward (last 100 episodes) = 3.34
Episode 7300: Average Reward (last 100 episodes) = 3.6
Episode 7400: Average Reward (last 100 episodes) = 5.48
Episode 7500: Average Reward (last 100 episodes) = 3.92
Episode 7600: Average Reward (last 100 episodes) = 5.44
Episode 7700: Average Reward (last 100 episodes) = 5.34
Episode 7800: Average Reward (last 100 episodes) = 5.19
Episode 7900: Average Reward (last 100 episodes) = 5.1
Episode 8000: Average Reward (last 100 episodes) = 4.07
Episode 8100: Average Reward (last 100 episodes) = 4.79
Episode 8200: Average Reward (last 100 episodes) = 5.87
Episode 8300: Average Reward (last 100 episodes) = 5.3
Episode 8400: Average Reward (last 100 episodes) = 4.44
Episode 8500: Average Reward (last 100 episodes) = 5.18
Episode 8600: Average Reward (last 100 episodes) = 4.77
Episode 8700: Average Reward (last 100 episodes) = 3.83
Episode 8800: Average Reward (last 100 episodes) = 4.78
Episode 8900: Average Reward (last 100 episodes) = 4.63
Episode 9000: Average Reward (last 100 episodes) = 5.51
Episode 9100: Average Reward (last 100 episodes) = 4.4
Episode 9200: Average Reward (last 100 episodes) = 4.57
Episode 9300: Average Reward (last 100 episodes) = 3.68
Episode 9400: Average Reward (last 100 episodes) = 3.88
Episode 9500: Average Reward (last 100 episodes) = 4.32
Episode 9600: Average Reward (last 100 episodes) = 4.13
Episode 9700: Average Reward (last 100 episodes) = 5.74
Episode 9800: Average Reward (last 100 episodes) = 4.06
Episode 9900: Average Reward (last 100 episodes) = 5.13
Episode 10000: Average Reward (last 100 episodes) = 4.97
Training Complete
```

DQN OUTPUT



0
50
100
150
200
250
300
0
100
200
300
400
500



Deep Q-Learning on Taxi-v3: Reward vs Episode
Reward
Episode
0
−100
−200
−300
−400
−500
−600
0
2000
4000
6000
8000
10000

RESULTS AND OBSERVATIONS

Q-Learning

✅ Fast convergence (~7.5 avg. reward) ⚠️ Slight instability in early training.

SARSA

✅ More stable learning (~7.3-7.5 avg. reward) ⚠️ Slower convergence.

DQN

⚠️ Poor performance (~-1.02 avg. reward) ⚠️ Unstable learning, requires tuning.

Value Iteration

✅ Fastest convergence (~optimal policy achieved)
✅ Most stable, but high computation cost.

RESULTS AND OBSERVATIONS

Algorithm	Type	Convergence Speed	Stability	Final Performance
Q-Learning	Off-Policy	Fast	Moderate	~7.5
SARSA	On-Policy	Slower	More Stable	~7.3-7.5
DQN	Deep Learning	Slow	Least Stable	~5.4
Value Iteration	DP-Based	Fastest	Most Stable	Optimal

KEY TAKEAWAYS

1. Value Iteration is best for small environments (fast, stable, optimal policy).
2. Q-Learning is fast but slightly unstable (good for quick learning).
3. SARSA is slower but stable (avoids aggressive actions).
4. DQN is ineffective in small discrete environments (better suited for complex tasks).

CONCLUSION

- Successfully trained an autonomous taxi agent using RL.
- Compared and analyzed Q-Learning, SARSA, DQN, and Value Iteration.
- Identified strengths, weaknesses, and best use cases for each algorithm

THANK YOU

QUESTIONS?