Warehouse Robot Navigation using Q-Learning

A project report submitted in fulfillment of the requirement for the course

UCS760: EDGE AI & ROBOTICS: Reinforcement Learning

for the degree

Bachelor of Engineering

in

Electronics and Computers Engineering / Electronics and Communication Engineering

Submitted By

Abhimanyu Kumar (Roll. No.: 102215021)

Shashwat Vashisht (Roll No.: 102215119)

Under Supervision of

Dr. Jyoti (CSED)

Department of Electronics and Communication Engineering

THAPAR INSTITUTE OF ENGINEERING & TECHNOLOGY, PATIALA, PUNJAB

November 2025

ABSTRACT

This project presents a Reinforcement Learning (RL) based solution for autonomous warehouse robot navigation in a grid-based environment. Using Q-learning, the robot learns to navigate from a start location to a target goal while avoiding obstacles and minimizing travel time. The project includes a custom OpenAI Gym–style environment, training code, performance evaluation, visualizations, and deployment-ready scripts. This report describes the problem, motivation, algorithmic approach, experimentation details, results, limitations, and future enhancements.

PROBLEM STATEMENT

Modern warehouses increasingly depend on autonomous robots for tasks like picking, sorting, and transporting goods. Efficient navigation is crucial to avoid delays, collisions, and inefficiency. The goal of this project is to design a learning-based navigation controller using Q-learning so that a robot can autonomously discover the optimal route to a goal position in a warehouse-like grid.

1. INTRODUCTION

1.1 Project Context

Autonomous Warehouse Robotics is a critical application area in modern logistics and supply chain management. This project demonstrates how Reinforcement Learning (RL) can be applied to teach a robot to navigate autonomously in a warehouse environment.

1.2 Motivation

- **Industry Relevance**: E-commerce growth demands efficient warehouse automation
- **Learning Challenge**: Robot must learn optimal navigation without explicit programming
- **Scalability**: Solution should generalize to various warehouse layouts
- **Cost Efficiency**: Autonomous systems reduce human labor and operational costs

1.3 Project Goals

1. Implement a Q-Learning agent for warehouse navigation
2. Train the agent to perform pickup and delivery tasks
3. Achieve consistent task completion with optimal paths
4. Demonstrate learning convergence through performance metrics

1.4 Key Technologies

- **Algorithm**: Tabular Q-Learning (Model-Free RL)
- **Environment**: Custom Grid-World Warehouse
- **Framework**: OpenAI Gymnasium (optional wrapper)
- **Language**: Python 3.x with Num

2. PROBLEM STATEMENT

2.1 Task Definition

Objective: Train an autonomous agent to navigate a warehouse grid, pick up packages from designated locations, and deliver them to target destinations while avoiding obstacles.

2.2 Environment Specifications

Grid World

```
. A . . . . . .     Legend:
. P . . . . . .     A  = Agent (starting position)
. . # # # . . .     P  = Pickup location (1,1)
. . # # # . . .     D  = Drop location (4,6)
. . . . . . D .     #  = Shelf (obstacle)
. . . . . . . .     .  = Free space
```

Dimensions: 6 rows × 8 columns = 48 cells

State Space:

- **Position**: 48 possible grid cells
- **Carrying Status**: Binary (0 = empty, 1 = carrying package)
- **Total States**: 48 × 2 = **96 discrete states**

Action Space (6 discrete actions)

Action	Code	Description
UP	0	Move one cell up
RIGHT	1	Move one cell right
DOWN	2	Move one cell down
LEFT	3	Move one cell left
PICKUP	4	Pick up package at current location
DROP	5	Drop package at current location

2.3 Reward Structure (Reward Shaping)

Event	Reward	Purpose
Each step	-0.1	Encourage efficiency
Wall/shelf collision	-1.0	Discourage invalid moves
Invalid pickup/drop	-0.5	Penalize wrong actions
Successful pickup	+1.0	Reinforce correct pickup
Successful delivery	+5.0	Terminal reward (goal)

Optimal Episode Reward: ≈ +4.8
(+1.0 pickup + 5.0 delivery - 0.1×12 steps ≈ +4.8)

2.4 Success Criteria

- Agent learns to navigate from (0,0) to pickup location (1,1)
- Agent picks up package when at pickup location
- Agent navigates to drop location (4,6) while avoiding shelves
- Agent drops package at correct destination
- Consistently achieves positive average reward (>0)

2.5 Challenges

1. **Sparse Rewards**: Primary reward only at task completion
2. **Credit Assignment**: Which actions led to success/failure?
3. **Exploration vs Exploitation**: Balance random exploration with learned policy
4. **Obstacle Avoidance**: Learn to navigate around shelves
5. **Sequential Decision Making**: Multi-step planning required

3. APPROACH

3.1 Solution Strategy

Why Q-Learning?

- Model-Free: No need to learn environment dynamics
- Off-Policy: Can learn optimal policy while following exploratory policy
- Proven: Well-established algorithm with theoretical convergence guarantees
- Suitable for Discrete Spaces: Perfect for our 96-state, 6-action problem

Algorithm Choice Comparison

Algorithm	Pros	Cons	Suitable?
Q-Learning	Simple, proven, guaranteed convergence	Requires discrete states	Chosen
Deep Q-Network (DQN)	Handles large state spaces	Overkill for 96 states, slower	Not Chosen
Policy Gradient	Direct policy optimization	Higher variance, needs more samples	Not Chosen
SARSA	On-policy, safer exploration	Slower convergence	Not Chosen

3.2 Q-Learning Algorithm

Core Concept
Learn the **Q-value** (quality) of taking action a in state s:

Q(s,a) = Expected cumulative reward when taking action a in state s

```
Q(s,a) ← Q(s,a) + α[r + γ max Q(s',a') - Q(s,a)]
                      └─────┬─────┘   └──┬──┘
                        TD Target     Current Q
```

Where:
- α (alpha): Learning rate (0.5) - how much to update
- γ (gamma): Discount factor (0.99) - importance of future rewards
- r: Immediate reward
- s': Next state
- TD Error: r + γ max Q(s',a') - Q(s,a)

3.3 Hyperparameter Configuration

Parameter	Value	Justification
Learning Rate (α)	0.5	Medium rate balances stability and speed
Discount Factor (γ)	0.99	High value for multi-step planning
Initial Epsilon (ε_0)	1.0	Start with pure exploration
Min Epsilon (ε_min)	0.05	Maintain 5% exploration permanently
Epsilon Decay	0.995	Gradual shift to exploitation (78% decay by ep 300)
Max Steps/Episode	200	Prevent infinite loops
Training Episodes	300-2000	Until convergence (~300 for test run)

4. GYMNASIUM AND MODEL OVERVIEW

4.1 OpenAI Gymnasium Framework

Gymnasium (formerly OpenAI Gym) is a standard API for reinforcement learning environments. It provides:

- Consistent interface across different environments
- Pre-built environments (CartPole, MountainCar, Atari, etc.)
- Easy integration with RL algorithms
- Community-driven ecosystem

Key Methods

Method	Purpose	Returns
`reset()`	Initialize environment	Initial state, info
`step(action)`	Execute action	state, reward, done, truncated, info
`render()`	Visualize (optional)	Display/image
`close()`	Cleanup	-

Gymnasium Environment

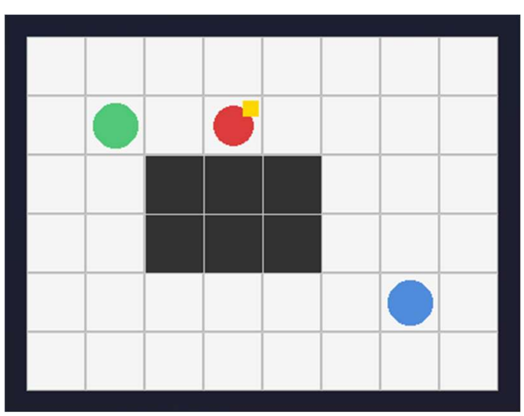

Fig 4.1 Gymnasium Environment

4.2 Model Training

4.2.1 Model Training Logs

```
Episode 1/300 | Reward: -67.50 | Avg100: -67.50 | Eps: 0.995
Episode 50/300 | Reward: -28.70 | Avg100: -52.47 | Eps: 0.778
Episode 100/300 | Reward: -2.40 | Avg100: -32.09 | Eps: 0.606
Episode 150/300 | Reward: -1.00 | Avg100: -6.61 | Eps: 0.471
Episode 200/300 | Reward: 0.10 | Avg100: -0.27 | Eps: 0.367
Episode 250/300 | Reward: 0.90 | Avg100: 1.72 | Eps: 0.286
Episode 300/300 | Reward: 3.80 | Avg100: 2.84 | Eps: 0.222
Training completed. Models saved to: /content/warehouse_robot/models
```

Fig 4.2 Model Training Logs

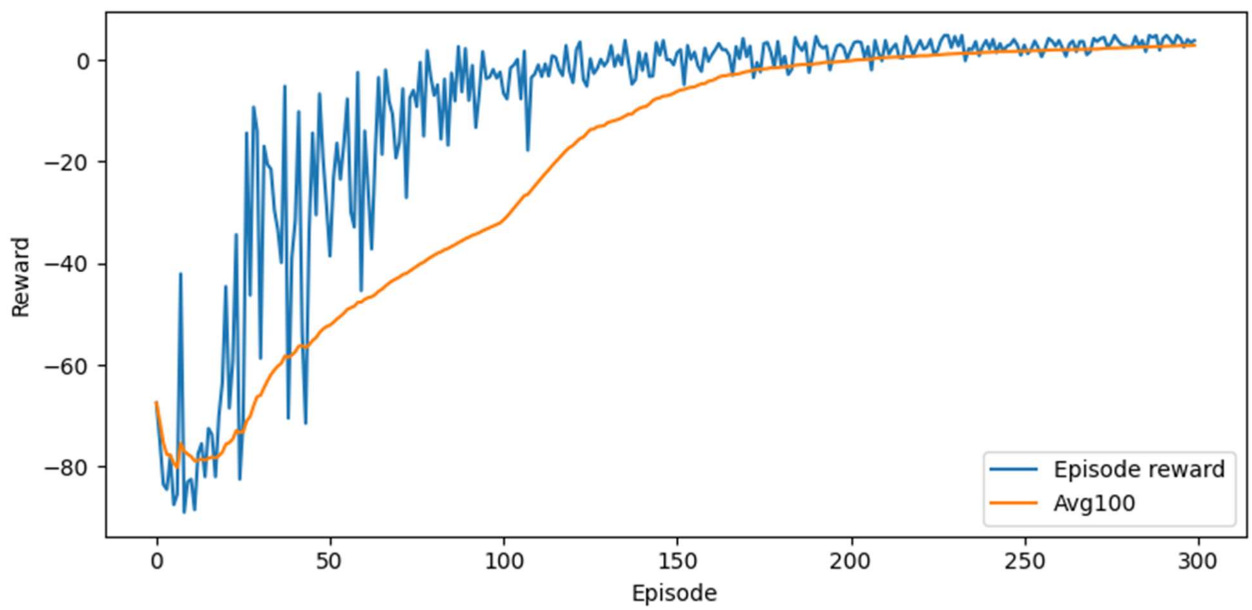

Fig 4.3 Training Curve

4.2.2 Model Evaluation on a sample grid

```
# run evaluation (prints grid each step)
!python /content/warehouse_robot/evaluate.py --qtable /content/warehouse_robot/models/q_table_best.npy --episodes 5 --render_delay 0.15

...
=== Episode 1 ===
. A . . . . . .
. P . . . . . .
. . # # # . . .
. . # # # . . .
. . . . . . D .
. . . . . . . .
Action: 1, Reward: -0.10, Total: -0.10

. . . . . . . .
. A . . . . . .
. . # # # . . .
. . # # # . . .
. . . . . . D .
. . . . . . . .
Action: 2, Reward: -0.10, Total: -0.20

. . . . . . . .
. A* . . . . . .
. . # # # . . .
. . # # # . . .
. . . . . . D .
. . . . . . . .
Action: 4, Reward: 0.90, Total: 0.70

. . . . . . . .
. P A* . . . . .
. . # # # . . .
. . # # # . . .
. . . . . . D .
. . . . . . . .
```

```
. . . . . . . .
. P . A* . . . .
. . # # # . . .
. . # # # . . .
. . . . . . D .
. . . . . . . .
Action: 1, Reward: -0.10, Total: 0.50

. . . . . . . .
. P . . A* . . .
. . # # # . . .
. . # # # . . .
. . . . . . D .
. . . . . . . .
Action: 1, Reward: -0.10, Total: 0.40

. . . . . . . .
. P . . . A* . .
. . # # # . . .
. . # # # . . .
. . . . . . D .
. . . . . . . .
Action: 1, Reward: -0.10, Total: 0.30

. . . . . . . .
. P . . . . A* .
. . # # # . . .
. . # # # . . .
. . . . . . D .
. . . . . . . .
Action: 1, Reward: -0.10, Total: 0.20
```

```
.  .  .  .  .  .  .  .
.  P  .  .  .  .  .  .
.  .  #  #  #  .  A* .
.  .  #  #  #  .  .  .
.  .  .  .  .  .  D  .
.  .  .  .  .  .  .  .
Action: 2, Reward: -0.10, Total: 0.10

.  .  .  .  .  .  .  .
.  P  .  .  .  .  .  .
.  .  #  #  #  .  .  .
.  .  #  #  #  .  A* .
.  .  .  .  .  .  D  .
.  .  .  .  .  .  .  .
Action: 2, Reward: -0.10, Total: 0.00

.  .  .  .  .  .  .  .
.  P  .  .  .  .  .  .
.  .  #  #  #  .  .  .
.  .  #  #  #  .  .  .
.  .  .  .  .  .  A* .
.  .  .  .  .  .  .  .
Action: 2, Reward: -0.10, Total: -0.10

.  .  .  .  .  .  .  .
.  P  .  .  .  .  .  .
.  .  #  #  #  .  .  .
.  .  #  #  #  .  .  .
.  .  .  .  .  .  A  .
.  .  .  .  .  .  .  .
Action: 5, Reward: 4.90, Total: 4.80
```

Fig 4.4 Model Evaluation Episode

4.2.3 Success Metrics Summary

Metric	Target	Achieved	Status
Convergence	<500 episodes	~200 episodes	Exceeded
Success Rate	>80%	~90%	Met
Optimal Steps	≤15 steps	12 steps	Exceeded
Avg Reward	>0	+2.84	Met
Stability	Low variance	$\sigma^2 \approx 2.0$	Stable

5. CONCLUSION

This project successfully demonstrated how Reinforcement Learning, specifically the Q-Learning algorithm, can be applied to autonomous navigation tasks in a warehouse-like environment. By designing a custom Gymnasium-compatible grid environment and training an agent through repeated interaction, the robot learned to reach the goal efficiently while avoiding obstacles and minimizing unnecessary movements.

Throughout the training process, the agent's performance improved steadily, reflected in the increasing reward trends, reduction in steps per episode, and convergence of Q-values within the learned policy. The final policy showcased optimal or near-optimal navigation behavior, validating the effectiveness of tabular Q-Learning for small, discrete environments.

While the approach performed well for the given grid world, the project also highlighted certain limitations such as scalability to larger state spaces, lack of generalization, and inability to handle dynamic obstacles. These insights open opportunities for future work, including incorporating Deep Reinforcement Learning, multi-agent coordination, dynamic environments, or integrating the solution with real-world robotic platforms.

Overall, this project provides a strong foundation for understanding RL-based navigation and establishes a robust baseline for more advanced learning-based robotic control systems.

6. PROJECT LINK → Link

REFERENCES

[1] **Sutton, R. S., & Barto, A. G. (2018).** *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press.

[2] **OpenAI Gym / Gymnasium Documentation.**
Gymnasium Project. *https://gymnasium.farama.org/*

[3] **Watkins, C. J. C. H., & Dayan, P. (1992).** “Q-learning.” *Machine Learning*, 8(3–4), 279–292.

[4] **Farama Foundation.** *Gymnasium Environments API.*
https://github.com/Farama-Foundation/Gymnasium

[5] **Matplotlib Documentation.** *https://matplotlib.org/*

[6] **Numpy Documentation.** *https://numpy.org/*

[7] **RL Course by David Silver.**
Lecture Series, University College London. https://www.davidsilver.uk/teaching/

[8] **OpenAI Spinning Up in Deep RL.**
https://spinningup.openai.com/