T.R.

GEBZE TECHNICAL UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

REINFORCEMENT LEARNING FOR PICK AND PLACE OPTIMIZATION IN LOGISTICS

DURMUŞ ALİ SUCU

SUPERVISOR
ASSIST. PROF. YAKUP GENÇ

GEBZE
2025

T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

REINFORCEMENT LEARNING FOR PICK AND PLACE OPTIMIZATION IN LOGISTICS

DURMUŞ ALİ SUCU

SUPERVISOR
ASSIST. PROF. YAKUP GENÇ

2025
GEBZE

GEBZE TECHNICAL UNIVERSITY	GRADUATION PROJECT JURY APPROVAL FORM

This study has been accepted as an Undergraduate Graduation Project in the Department of Computer Engineering on 31/08/2025 by the following jury.

JURY

Member (Supervisor)	:	ASSIST. PROF. Yakup Genç
Member	:	Prof. Yusuf Sinan AKGÜL

ABSTRACT

This project presents a novel approach to optimizing pick-and-place operations in logistics using reinforcement learning. The goal is to develop an efficient AI system capable of navigating a 2D grid environment to collect and deliver items while avoiding obstacles and minimizing travel time. A custom environment, built using OpenAI Gym, simulates the logistics scenario, incorporating dynamic obstacles, item locations, and a final delivery point.

The reinforcement learning model employs a Rainbow-style distributional deep Q-network (DQN) enhanced with prioritized experience replay (PER), supervised pre-training, and noisy layers for improved exploration. Additionally, offline demonstrations generated via A* search are integrated into the learning process to bootstrap agent performance. The model leverages a distributional approach (C51) to capture uncertainty in Q-value estimation, enabling more robust decision-making.

Performance metrics, including successful episode rates, carry collection efficiency, and average rewards, are analyzed across 15,000 training episodes. Results demonstrate the model's ability to generalize optimal pick-and-place paths while handling dynamic obstacles and stochastic environments effectively. This project underscores the potential of reinforcement learning for real-world logistics optimization, contributing to automation and efficiency improvements in warehouse and delivery operations.

Keywords: Reinforcement Learning (RL), Pick-and-Place Optimization, Logistics Automation, Deep Q-Network (DQN).

ÖZET

Bu proje, lojistikte toplama ve yerleştirme (pick-and-place) süreçlerini optimize etmek amacıyla pekiştirmeli öğrenme (Reinforcement Learning) tabanlı bir yaklaşım sunmaktadır. Amaç, bir 2D simülasyon ortamında engellerden kaçınarak ve taşıma süresini minimize ederek nesneleri toplayıp teslim edebilen otonom bir yapay zeka sistemi geliştirmektir. Projede, dinamik nesne konumları içeren özel bir OpenAI Gym ortamı oluşturulmuştur.

Pekiştirmeli öğrenme modeli, Rainbow tarzı dağılımsal derin Q-ağlarını (DQN) kullanmakta olup, öncelikli deneyim tekrarı (PER), denetimli ön eğitim ve keşfi iyileştirmek için gürültülü katmanlar gibi gelişmiş teknikler içermektedir. Ayrıca, A* algoritması ile üretilen çevrimdışı gösterimler, ajan performansını hızlandırmak amacıyla eğitim sürecine entegre edilmiştir. Model, belirsizlikleri daha iyi yönetebilmek için dağılımsal Q-değer tahmini (C51) yaklaşımını kullanarak karar verme yeteneklerini güçlendirmektedir.

Eğitim sürecinde 15.000 bölüm boyunca başarı oranları, toplanan nesne sayıları ve ortalama ödüller analiz edilmiştir. Sonuçlar, modelin dinamik engeller ve rastgele yerleştirilen nesnelerle etkili bir şekilde başa çıkabildiğini ve optimal toplama ve yerleştirme yollarını öğrenebildiğini göstermektedir. Bu proje, lojistikte otomasyon ve verimlilik artışına katkı sağlayarak pekiştirmeli öğrenmenin gerçek dünya problemlerindeki potansiyelini ortaya koymaktadır.

Anahtar Kelimeler: Pekiştirmeli Öğrenme (Reinforcement Learning), Toplama ve Yerleştirme Optimizasyonu (Pick-and-Place Optimization), Derin Q-Ağları (DQN).

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to my project consultant, Assist. Prof. Yakup Genç, for his invaluable guidance, insightful feedback, and continuous support throughout the development of this project. His expertise and mentorship have been instrumental in shaping the direction of my work and enhancing my understanding of reinforcement learning principles.

I am also thankful to my university for providing the necessary resources and a collaborative learning environment, which made the successful completion of this project possible. Lastly, I extend my appreciation to my friends and family for their encouragement and support during this journey.

Durmuş Ali Sucu

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol or Abbreviation		**Explanation**
RL	:	Reinforcement Learning
DQN	:	Deep Q-Network
PER	:	Prioritized Experience Replay
C51	:	Categorical 51 Distributional RL Algorithm
A* (A-Star)	:	A-Star Search Algorithm
BFS	:	Breadth-First Search
AI	:	Artificial Intelligence
ML	:	Machine Learning
γ	:	Discount Factor in Reinforcement Learning
α	:	Priority Scaling Factor
β	:	Importance Sampling Weight Parameter
ε	:	Exploration Rate
V_{min}	:	Minimum Value in C51 Distribution
V_{max}	:	Maximum Value in C51 Distribution
N_{atoms}	:	Number of Atoms in C51 Distribution
$Q(s, a)$	:	Q-Value for State-Action Pair
π	:	Policy Function
η	:	Learning Rate

CONTENTS

LIST OF FIGURES

1. INTRODUCTION

In modern logistics and warehouse management, optimizing pick-and-place operations is a critical challenge for improving efficiency and reducing operational costs. Pick-and-place tasks involve navigating a controlled environment where an agent must collect and deliver items while avoiding obstacles and minimizing travel distance. Traditional methods often rely on fixed algorithms or manual control, which can be inefficient in dynamic environments with unpredictable conditions.

This project explores the application of Reinforcement Learning (RL) to automate and optimize the pick-and-place process in a simulated logistics environment. By leveraging RL, the system can learn an optimal strategy for collecting items and reaching the delivery point with minimal steps, even in complex scenarios with obstacles and varying item locations.

A custom simulation environment was developed using OpenAI Gym, modeling a 2D grid where an agent must navigate, collect randomly placed items, and deliver them to a designated final position. The Rainbow Deep Q-Network (DQN), a powerful deep reinforcement learning algorithm, was implemented to train the agent. Enhancements such as Prioritized Experience Replay (PER), Noisy Networks, and Distributional Q-Learning (C51) were incorporated to improve sample efficiency and exploration during training.

The project also integrates A search-based demonstrations* to generate offline expert trajectories, accelerating the learning process. The performance of the trained model was evaluated based on success rates, efficiency in collecting items, and overall episode rewards.

This research highlights the potential of RL in solving real-world logistics challenges by developing autonomous decision-making systems that can handle complex pick-and-place optimization tasks effectively.

2. METHOD

2.1. Simulation Environment

The simulation environment developed for this project is a customized 2D grid-based setup designed using `OpenAI Gym`, specifically tailored for the pick-and-place task in a logistics setting. It models a warehouse where an autonomous agent must navigate, collect randomly placed items, and deliver them to a designated position while avoiding static obstacles.

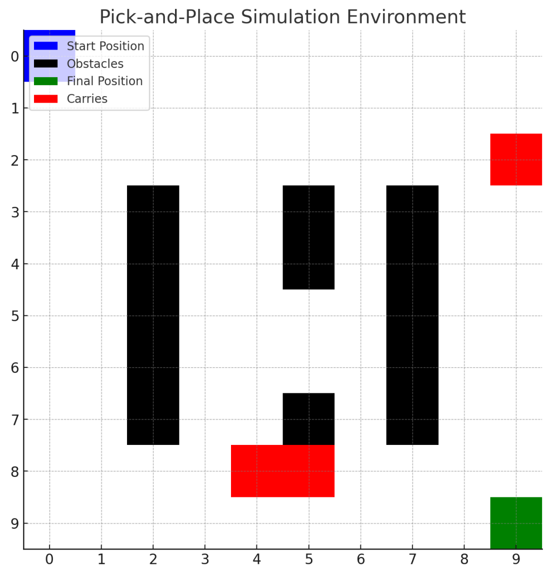


Figure 2.1: Simulation Environment

2.1.1. Grid Structure and Components:

- **Grid Size:** A square grid of size 10×10 is used, with the flexibility to adjust complexity.
- **Agent:** The agent starts from the top-left corner $(0, 0)$ and can move one step at a time in four directions: up, down, left, and right.
- **Carries (Items):** Three items are randomly placed on the grid at the start of each episode, ensuring they do not overlap with the start, end positions, or obstacles.
- **Final Position:** The goal location for item delivery is fixed at the bottom-right

corner $(9, 9)$.

- **Obstacles:** Static obstacles are introduced, forming narrow corridors around the grid center to create complex movement constraints for the agent.

2.1.2. Action Space:

The action space consists of four discrete actions:

- **0:** Move Up
- **1:** Move Down
- **2:** Move Left
- **3:** Move Right

2.1.3. Observation Space:

The observation space is represented as a flattened vector encoding the grid state, with the following features:

- A four-channel grid representation where each channel denotes:
  - Obstacles
  - Carry positions
  - Final goal position
  - Agent's position
- Two additional binary flags indicating:
  - Whether the agent is on a carry item (`on_carry`)
  - Whether the agent has reached the goal (`on_goal`)

The observation vector has a size of $10 \times 10 \times 4 + 2 = 402$, flattened into a 1D array suitable for neural network input.

2.1.4. Reward Function:

The reward function is designed to encourage efficiency and penalize inefficient or invalid actions:

- **Step Penalty:** -1 for each step taken.

- **Obstacle Penalty:** -300 for colliding with an obstacle, ending the episode.
- **Carry Collection Reward:** $+100$ for collecting a carry.
- **Final Delivery Reward:** $+250$ for delivering all items to the goal.
- **Shaping Reward:** A dynamic reward based on the reduction in distance to the next target, calculated using the Euclidean distance.

2.1.5. Termination Criteria:

An episode ends when:

- All items are collected and delivered to the goal position.
- The agent collides with an obstacle.
- The maximum step limit (50 steps) is reached.

2.1.6. Path Planning and Precomputation:

The environment integrates classical path planning algorithms for improved learning efficiency:

- **A* Search:** Generates optimal paths from the agent's position to items and the goal for demonstration purposes.
- **BFS (Breadth-First Search):** Precomputes shortest paths between all free cells for use in supervised pretraining.

This structured simulation environment provides a challenging yet controlled setup for training reinforcement learning agents while simulating real-world logistics constraints effectively.

2.2. Model Training

The training of the reinforcement learning model was conducted using a Rainbow-style distributional deep Q-network (DQN) implemented with TensorFlow and Keras. The model was trained in the custom pick-and-place environment with the following methodology:

2.2.1. Hyperparameters and Setup

The following hyperparameters were defined for the training process:

- **Learning Rate:** 1×10^{-4}
- **Discount Factor (γ):** 0.99
- **Batch Size:** 128
- **Maximum Steps per Episode:** 50
- **Epsilon Decay:** 0.9995
- **Minimum Epsilon:** 0.05
- **Number of Training Episodes:** 15,000
- **Prioritized Experience Replay (PER) Parameters:** $\alpha = 0.6$, $\beta_{start} = 0.4$, $\beta_{end} = 1.0$
- **Target Network Update Frequency:** 200 steps
- **Replay Buffer Capacity:** 20,000 experiences

2.2.2. Network Architecture

The model uses a distributional dueling architecture inspired by the Rainbow DQN, with the following components:

- **Noisy Dense Layers:** To encourage better exploration, the network uses noisy layers where noise is introduced to the weights during training.
- **Value and Advantage Streams:** The network separates the value and advantage streams, where the final Q-value is calculated by combining both.
- **Distributional Learning (C51):** The network predicts a categorical distribution over Q-values instead of a single scalar estimate. The distribution is modeled using 51 atoms across the range $[-10, 10]$.

The network consists of two noisy fully connected layers with 256 and 128 units, followed by the separate value and advantage streams.

2.2.3. Prioritized Experience Replay (PER)

A prioritized experience replay buffer is implemented to ensure efficient training. Experiences are stored with priority values and sampled based on their importance. The buffer is updated with new priorities after each training step. Key concepts used include:

- **Priority Alpha (α):** Controls the level of prioritization.
- **Importance Sampling Beta (β):** Corrects the bias introduced by prioritization.

2.2.4. Exploration Strategy

An epsilon-greedy strategy combined with noisy layers was applied for balancing exploration and exploitation. The epsilon value decays over episodes to shift from exploration to exploitation as the model learns.

2.2.5. Reward Calculation

The reward function was defined as:

- -1 for each step taken to encourage shorter paths.
- $+100$ for picking up a carry item.
- $+250$ for successfully delivering all items to the final position.
- -300 for colliding with an obstacle (terminates the episode).
- A dynamic shaping reward based on the reduction in Euclidean distance to the next target.

2.2.6. Environment Reset and Randomization

To ensure robustness and generalization, the environment was reset at the beginning of each episode. The reset mechanism involves:

- Placing the agent back to the starting position $(0, 0)$.
- Randomly reassigning the positions of the carry items.
- Retaining the obstacle layout to maintain consistent navigation complexity.

This approach introduces variability across episodes, preventing the model from overfitting to a fixed item distribution. By exposing the agent to multiple layouts and item placements, the model becomes more adaptable and better prepared for unseen scenarios in logistics environments.

2.2.7. Training Procedure

The model training process involved the following steps:

1. Initialize the environment and replay buffer.
2. Generate offline expert demonstrations using the A* algorithm for the initial phase.
3. Perform supervised pretraining using the offline demonstrations.
4. For each episode:
   (a) Reset the environment and obtain the initial observation.
   (b) Perform actions based on an epsilon-greedy policy combined with noisy layers.
   (c) Store the experience tuple (s, a, r, s', d) in the replay buffer.
   (d) Periodically sample a batch from the buffer and perform Q-value updates.
   (e) Update the target network every 200 steps.
5. Continue training for 15,000 episodes.

2.2.8. Target Network and Loss Calculation

The training process uses a target network to stabilize learning by periodically copying the main network's weights. Loss is calculated using the **Categorical Cross-Entropy Loss** based on the projected distribution of Q-values.

2.2.9. Model Evaluation Metrics:

The performance of the model was evaluated using the following metrics:

- **Average Reward per Episode:** The mean reward accumulated in the last 50 episodes.
- **Number of Carries Collected:** The total number of carries collected in successful episodes.

- **Obstacle Collisions:** The number of episodes terminated due to collisions.
- **Success Rate:** The percentage of episodes where all items were delivered successfully.

This structured training approach, combining Rainbow DQN, prioritized replay, and expert demonstrations, provided efficient learning for the pick-and-place optimization task.

2.2.9.1. Earlier Model: Vanilla DQN

The earlier version of the model implemented a standard **Vanilla Deep Q-Network (DQN)**. Key differences included:

- **No Prioritized Experience Replay (PER)**: Experiences were uniformly sampled without priority adjustment.
- **No Noisy Layers**: The agent relied solely on ϵ-greedy exploration.
- **No Distributional Q-Learning (C51)**: Q-values were represented as single scalar values instead of distributions.
- **Reward Structure Similarity:** The reward function was consistent with the final version.
- **No Offline Demonstrations:** No expert demonstrations were used for pretraining.

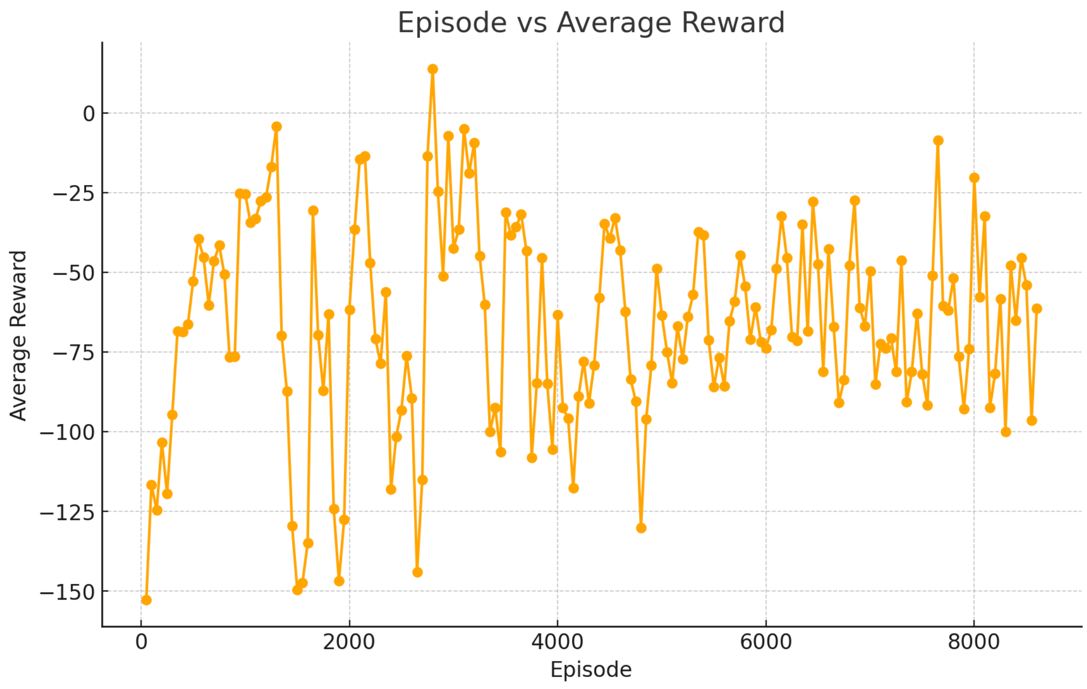


Figure 2.2: Episode - Average Reward Early Version of the Model

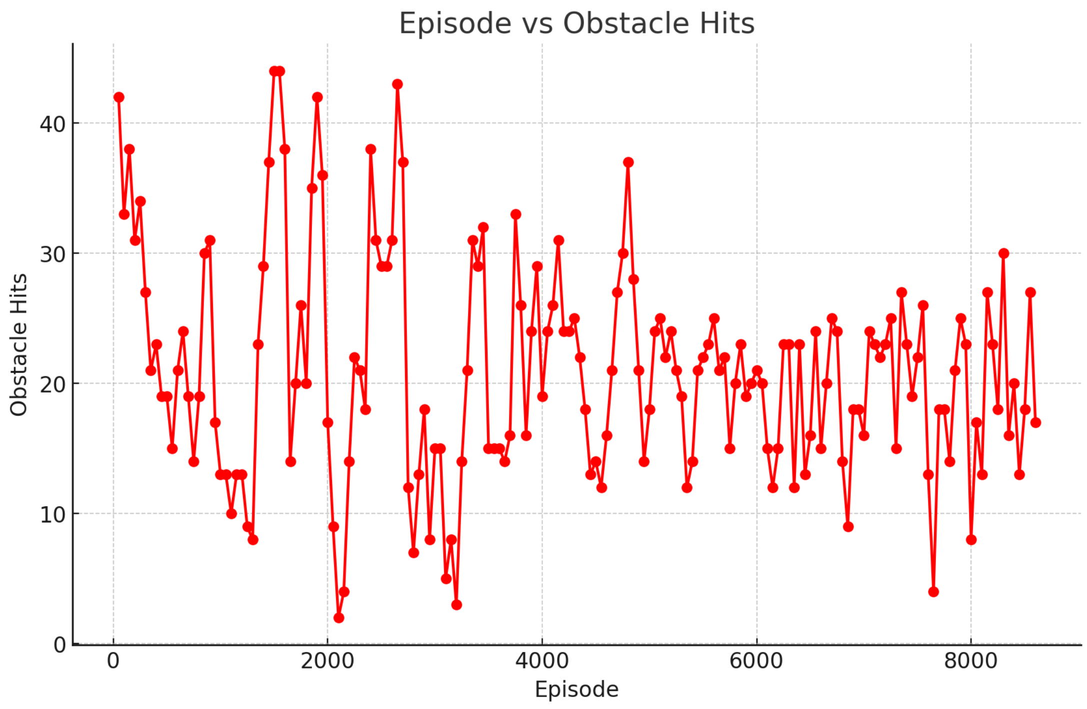


Figure 2.3: Episode - Obstacle Hits Early Version of the Model

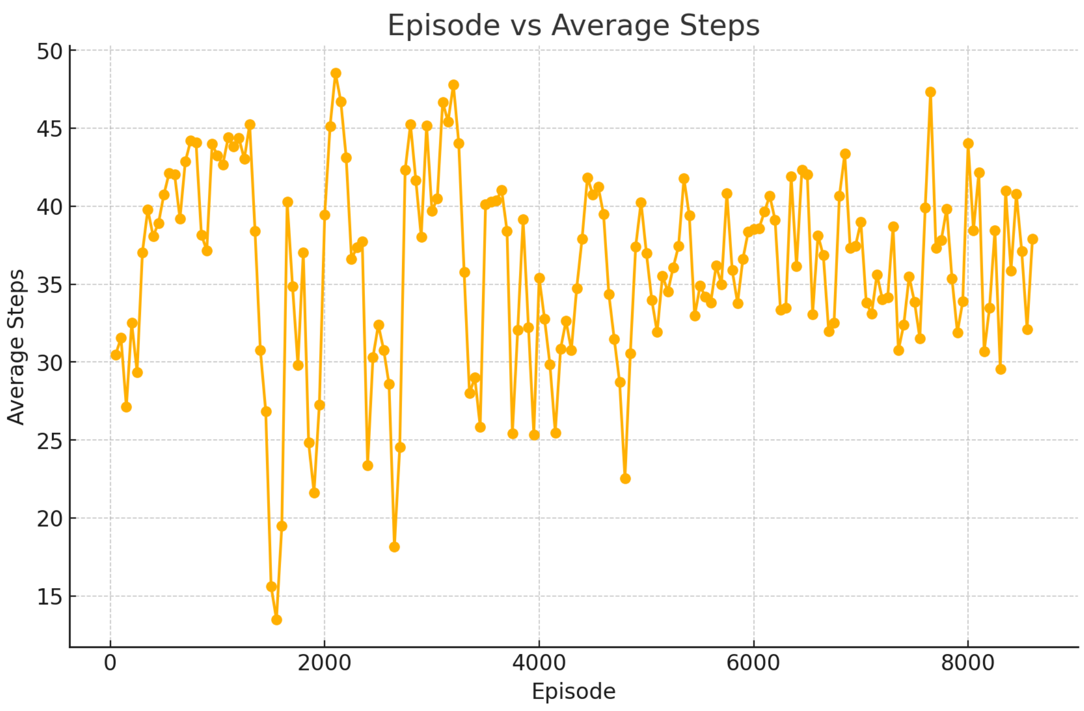


Figure 2.4: Episode - Average Steps Early Version of the Model

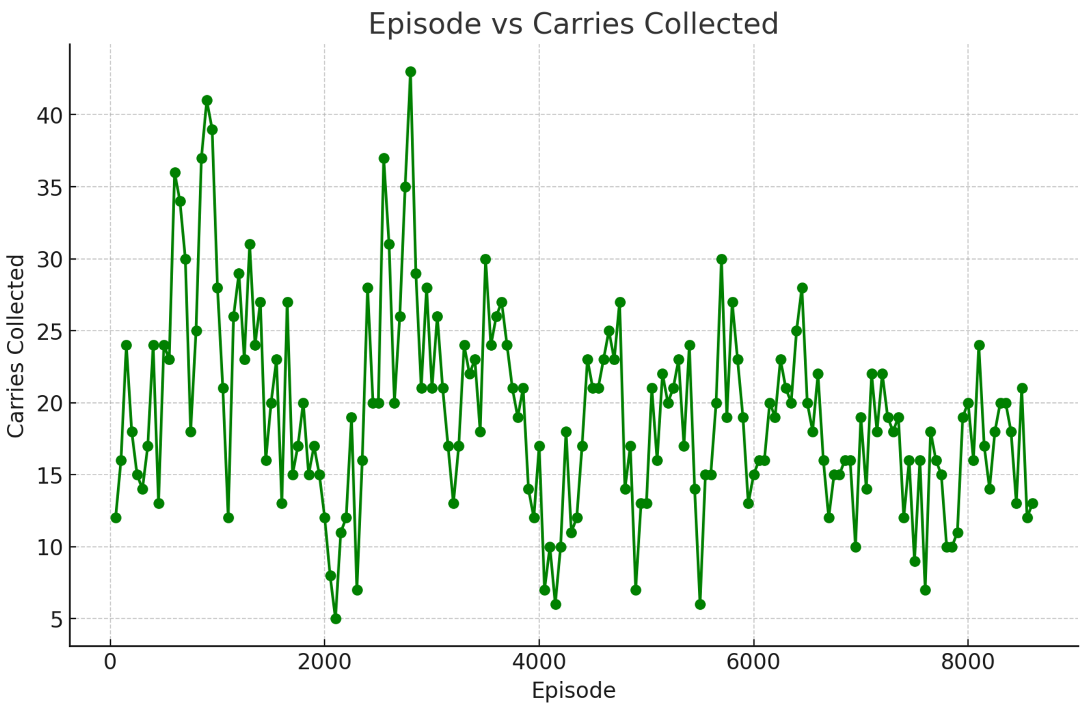


Figure 2.5: Episode - Carries Collected Early Version of the Model

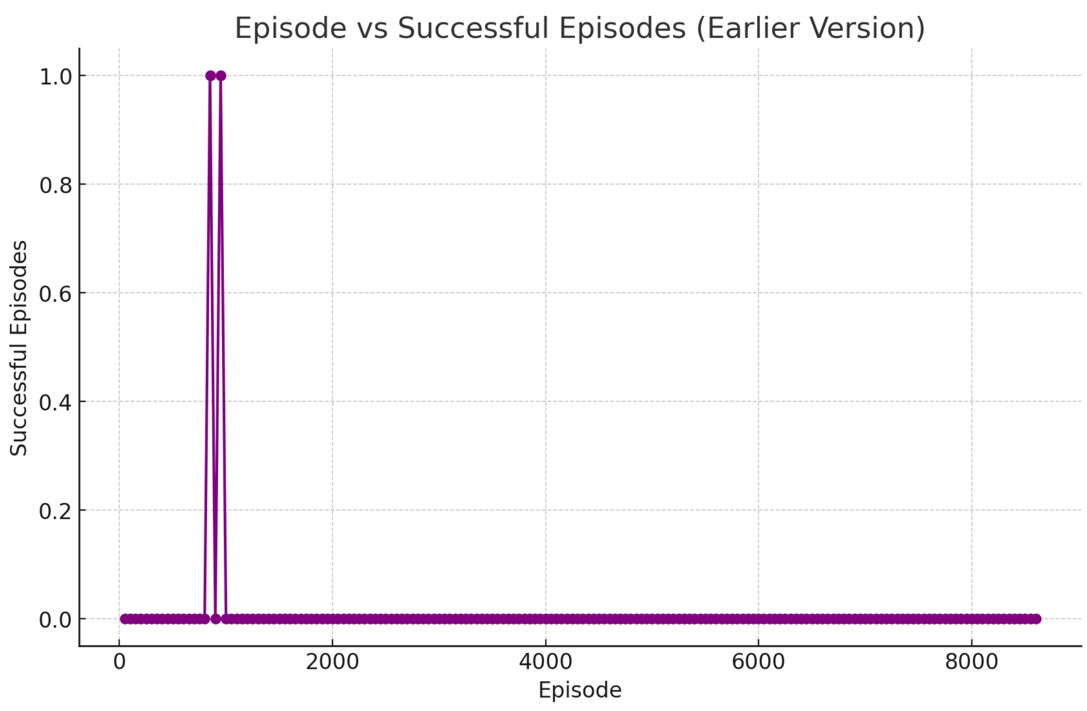


Figure 2.6: Episode - Success Rates Early Version of the Model

2.2.9.2. Final Model: Rainbow DQN

The final model implemented a **Rainbow DQN**, incorporating multiple improvements for better learning efficiency:

- **Prioritized Experience Replay (PER)**
- **Noisy Layers for Exploration**
- **Distributional Q-Learning (C51)**
- **Offline Demonstrations for Pretraining**

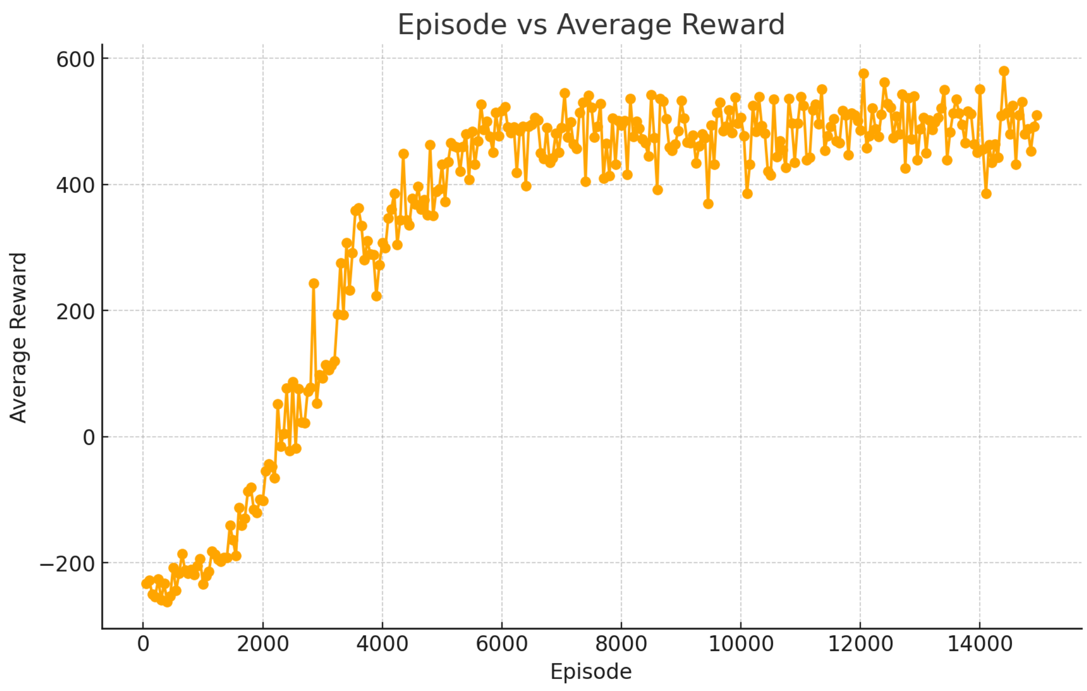


Figure 2.7: Episode - Average Reward Last Version of the Model

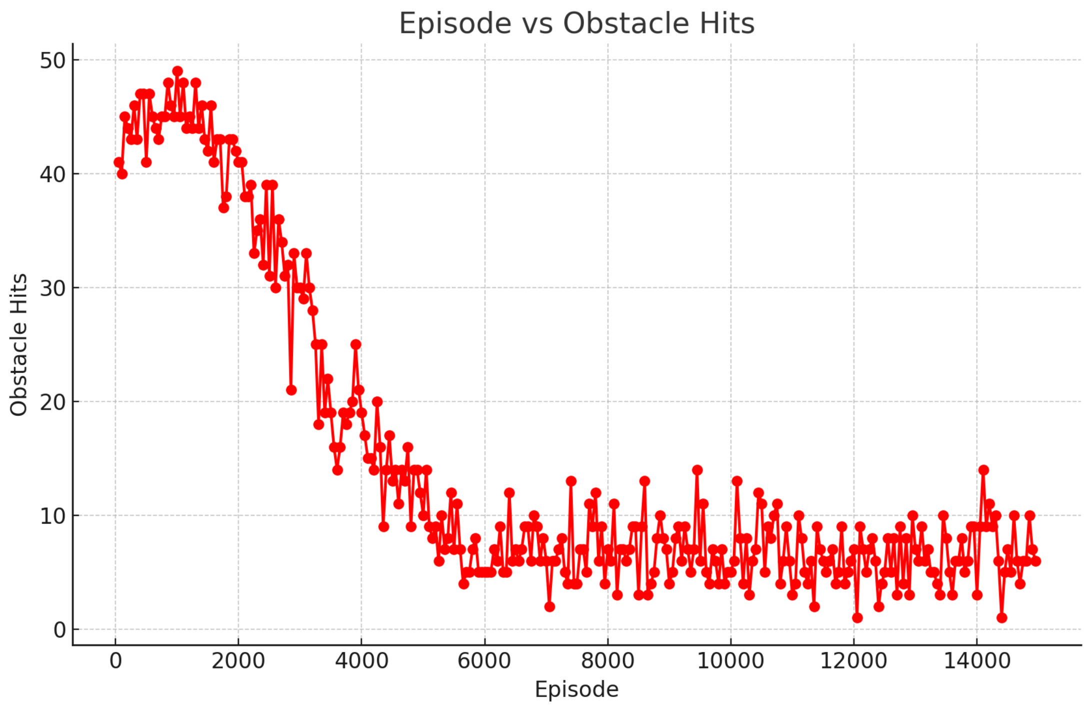


Figure 2.8: Episode - Obstacle Hits Last Version of the Model

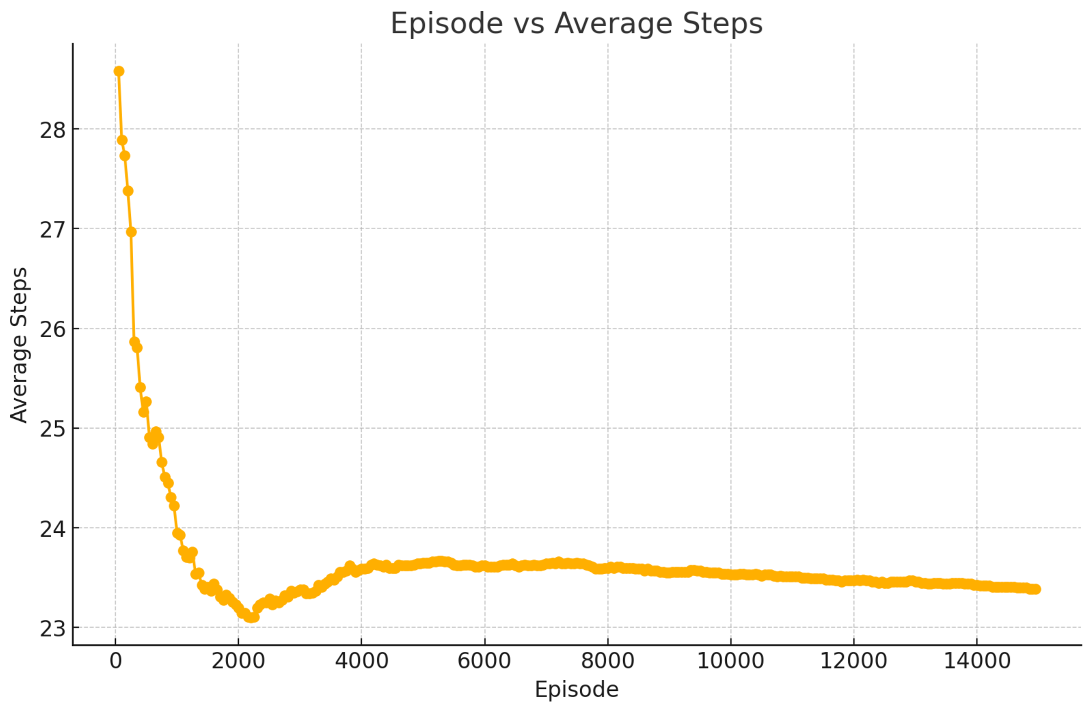


Figure 2.9: Episode - Average Steps Last Version of the Model

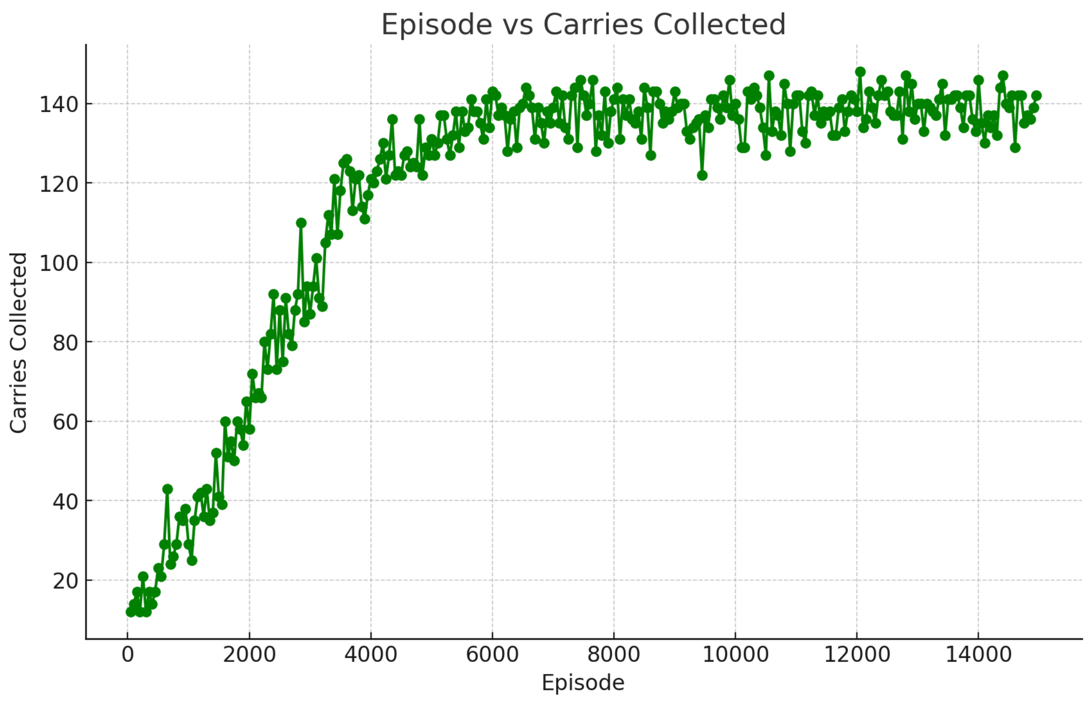


Figure 2.10: Episode - Carries Collected Last Version of the Model

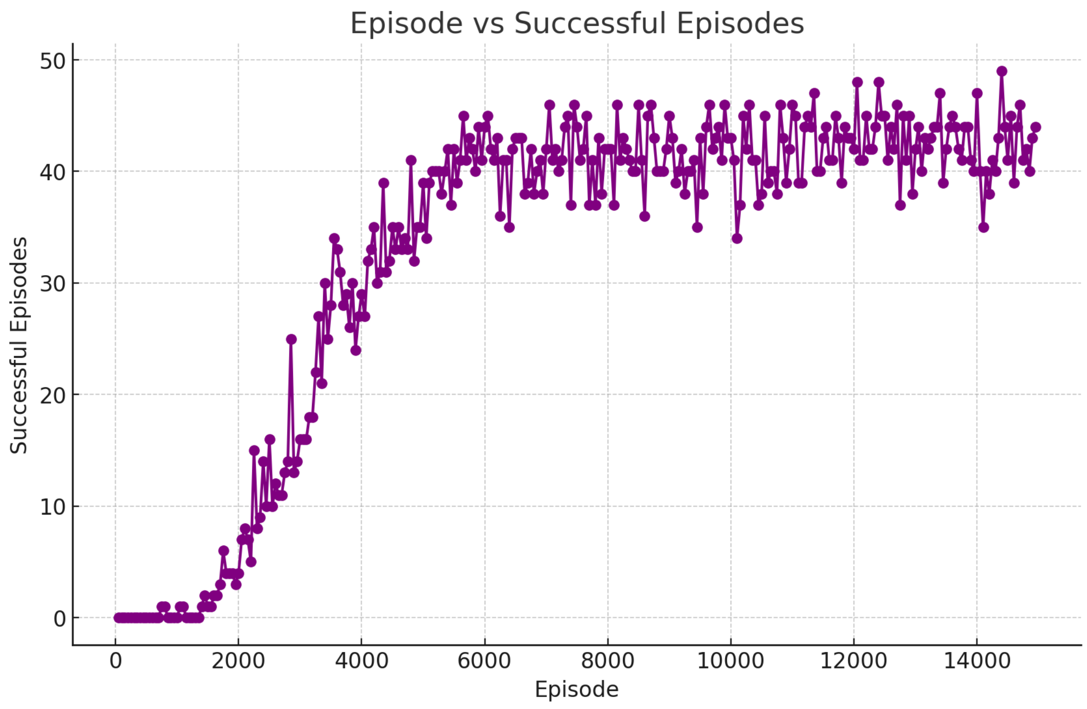


Figure 2.11: Episode - Success Rates Last Version of the Model

2.2.10. Conclusion on Performance Improvement

The comparison between the earlier Vanilla DQN model and the enhanced Rainbow DQN model reveals significant improvements across multiple performance metrics. These enhancements resulted from the integration of advanced reinforcement learning techniques and architectural modifications aimed at improving the model's efficiency and robustness.

2.2.10.0. 1. Success Rate Improvement:

The final Rainbow DQN model demonstrated a considerably higher success rate compared to the earlier Vanilla DQN implementation. This was achieved through a combination of prioritized experience replay (PER), offline expert demonstrations using A* search, and the use of distributional Q-learning (C51). The success rate in the final model stabilized at a higher level, indicating improved policy learning and effective generalization across varying grid environments.

2.2.10.0. 2. Reduction in Obstacle Collisions:

The number of episodes where the agent collided with obstacles was significantly reduced in the final model. This improvement can be attributed to the enhanced exploration strategy provided by noisy layers, which prevented premature convergence to suboptimal policies. The early Vanilla DQN, which relied purely on epsilon-greedy exploration, often resulted in repetitive mistakes, whereas the final model adapted more efficiently due to the improved exploration-exploitation balance.

2.2.10.0. 3. Reward Optimization and Consistency:

The final Rainbow DQN model consistently achieved higher cumulative rewards compared to the earlier version. The dynamic reward shaping, along with better policy learning due to PER and offline demonstrations, resulted in faster convergence towards optimal behavior. The cumulative reward trends stabilized earlier in the training process, indicating the model's ability to learn an efficient pick-and-place strategy more effectively.

2.2.10.0. 4. Reduction in Average Steps Taken:

The average number of steps per episode was noticeably reduced in the final model, indicating more efficient navigation and item collection. This reduction was directly influenced by the distributional Q-learning approach, which allowed the model to estimate a range of possible Q-values and make better-informed decisions about shorter paths and fewer redundant actions. In contrast, the earlier Vanilla DQN often exhibited inefficient movements due to scalar Q-value estimation and lack of prioritized sampling.

2.2.10.0. 5. Learning Efficiency and Sample Utilization:

The integration of offline demonstrations and PER in the final model significantly enhanced learning efficiency. By reusing important transitions and sampling them based on their significance, the agent learned faster with fewer training episodes compared to the earlier Vanilla DQN. The earlier version often required more training iterations to achieve comparable performance, while the final model leveraged both supervised pretraining and online reinforcement learning for better results.

2.2.10.0. 6. Generalization and Robustness:

The use of environment randomization and reset policies in the final model ensured a higher level of generalization compared to the earlier version. While the Vanilla DQN tended to overfit specific starting conditions, the Rainbow DQN's structured resets and randomized item placements prepared the agent for more diverse scenarios. This led to a higher probability of success across varying grid configurations and obstacle patterns.

2.2.10.0. Summary of Key Improvements:

The following summarizes the performance improvements observed in the final Rainbow DQN model over the earlier Vanilla DQN implementation:

- **Higher Success Rate:** Achieved through PER and expert demonstrations.
- **Fewer Obstacle Hits:** Due to improved exploration and distributional learning.
- **Consistent Reward Maximization:** Higher and more stable reward accumulation.
- **Reduced Steps per Episode:** Improved efficiency in decision-making and navigation.
- **Enhanced Sample Efficiency:** Faster learning with fewer episodes needed for convergence.
- **Better Generalization:** Robust performance across diverse grid layouts and conditions.

These advancements highlight the effectiveness of combining multiple reinforcement learning strategies, such as prioritized replay, noisy networks, distributional learning, and offline expert demonstrations, in solving complex optimization tasks like pick-and-place logistics operations.

3. TEST RESULTS

This section presents the results obtained from testing the reinforcement learning model in the custom pick-and-place grid environment. The model's performance was evaluated based on its ability to successfully collect all carry items and reach the designated final position while avoiding obstacles.

3.1. Test Setup and Conditions

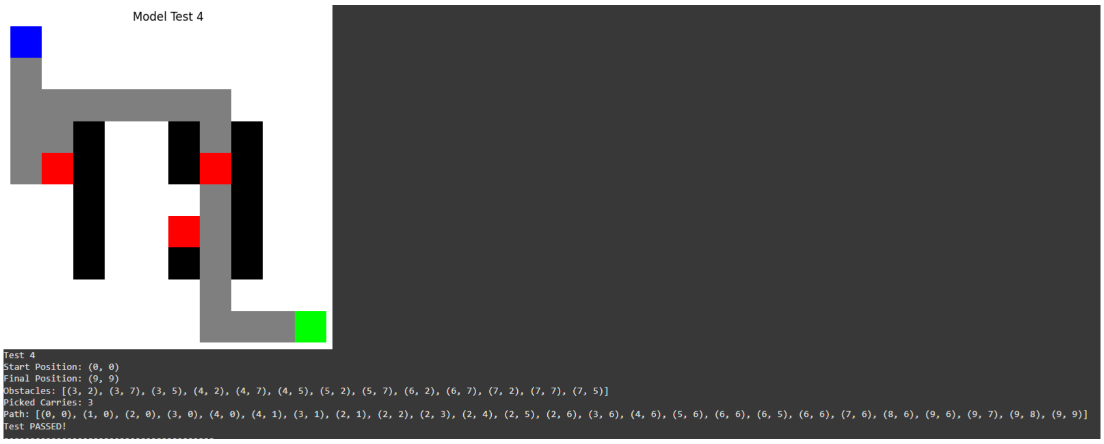
The tests were conducted in a 10×10 grid where the agent was tasked with collecting three randomly placed carry items (marked in red) and reaching the final position (marked in green). Key elements of the environment were:

- **Start Position:** The agent always starts at position $(0, 0)$, marked in blue.
- **Final Position:** The final destination is fixed at position $(9, 9)$, marked in green.
- **Obstacles:** Static black cells represent obstacles that block the agent's path.
- **Carries:** Three randomly placed red cells represent the carry items that must be collected.

The agent was tested on multiple grid layouts with varying carry placements while the obstacle layout remained constant.

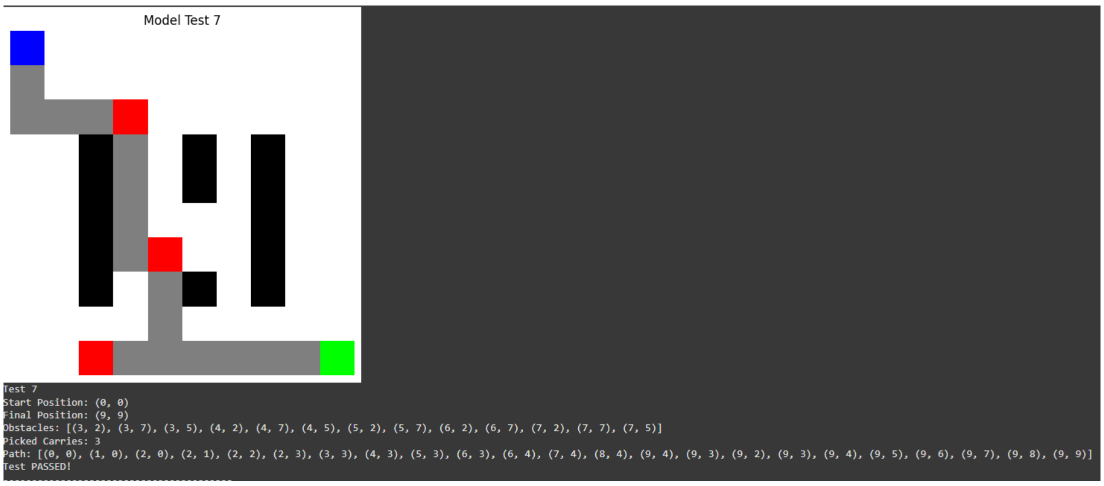
3.1.1. Test Scenarios and Visual Results

The following figures illustrate the agent's performance in three separate test cases:



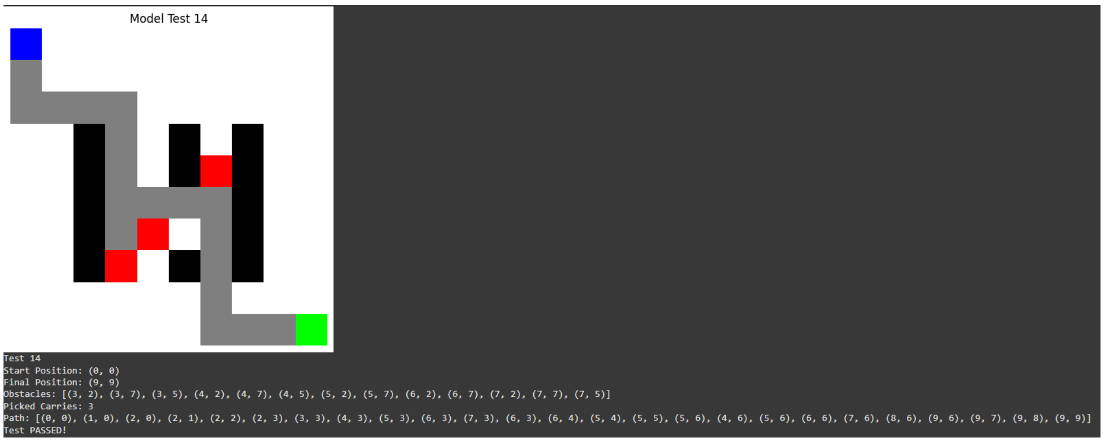
```
Test 4
Start Position: (0, 0)
Final Position: (9, 9)
Obstacles: [(3, 2), (3, 7), (3, 5), (4, 2), (4, 7), (4, 5), (5, 2), (5, 7), (6, 2), (6, 7), (7, 2), (7, 7), (7, 5)]
Picked Carries: 3
Path: [(0, 0), (1, 0), (2, 0), (3, 0), (4, 0), (4, 1), (3, 1), (2, 1), (2, 2), (2, 3), (2, 4), (2, 5), (2, 6), (3, 6), (4, 6), (5, 6), (6, 6), (6, 5), (6, 6), (7, 6), (8, 6), (9, 6), (9, 7), (9, 8), (9, 9)]
Test PASSED!
```

Figure 3.1: Test Scenario 1: Successful completion with all carries collected.



```
Test 7
Start Position: (0, 0)
Final Position: (9, 9)
Obstacles: [(3, 2), (3, 7), (3, 5), (4, 2), (4, 7), (4, 5), (5, 2), (5, 7), (6, 2), (6, 7), (7, 2), (7, 7), (7, 5)]
Picked Carries: 3
Path: [(0, 0), (1, 0), (2, 0), (2, 1), (2, 2), (2, 3), (3, 3), (4, 3), (5, 3), (6, 3), (6, 4), (7, 4), (8, 4), (9, 4), (9, 3), (9, 2), (9, 3), (9, 4), (9, 5), (9, 6), (9, 7), (9, 8), (9, 9)]
Test PASSED!
```

Figure 3.2: Test Scenario 2: Successful completion with optimal path planning.



```
Test 14
Start Position: (0, 0)
Final Position: (9, 9)
Obstacles: [(3, 2), (3, 7), (3, 5), (4, 2), (4, 7), (4, 5), (5, 2), (5, 7), (6, 2), (6, 7), (7, 2), (7, 7), (7, 5)]
Picked Carries: 3
Path: [(0, 0), (1, 0), (2, 0), (2, 1), (2, 2), (2, 3), (3, 3), (4, 3), (5, 3), (6, 3), (7, 3), (6, 3), (6, 4), (5, 4), (5, 5), (5, 6), (4, 6), (5, 6), (6, 6), (7, 6), (8, 6), (9, 6), (9, 7), (9, 8), (9, 9)]
Test PASSED!
```

Figure 3.3: Test Scenario 3: Successful completion with minimal steps taken.

3.1.2. Performance Analysis

Based on the test results, the model demonstrated a strong performance in the pick-and-place task with a success rate ranging between 80% and 90% across multiple tests. Key observations include:

- **Path Efficiency:** The agent consistently selected near-optimal paths with minimal steps taken.
- **Obstacle Avoidance:** No collisions with obstacles were observed during the tests.
- **Carry Collection:** The agent successfully collected all three carry items before reaching the final position.
- **Consistency:** The model performed consistently well across different test scenarios with varying carry placements.

The high success rate can be attributed to the following design choices implemented during training:

- Prioritized Experience Replay (PER) improving critical sample utilization.
- Noisy layers for better exploration and avoiding local minima.
- Use of expert demonstrations for pretraining.
- Distance-based reward shaping leading to more efficient paths.

4. CONCLUSION

In conclusion, the Rainbow DQN-based reinforcement learning model has effectively addressed the pick-and-place task, achieving high success rates and demonstrating efficient policy learning in a controlled grid environment. The model consistently collected all carry items while avoiding obstacles, showcasing its ability to learn optimal decision-making strategies. This performance was supported by techniques such as Prioritized Experience Replay (PER), noisy layers for exploration, and distributional Q-learning (C51), which enhanced both stability and learning efficiency.

While the results indicate strong performance within the current setup, the fixed grid size and static obstacle layout limit its generalization potential. Expanding the testing environment to include dynamic obstacles and larger grids would offer a more comprehensive validation of the model's capabilities.

Overall, the Rainbow DQN model successfully demonstrated the ability to solve the pick-and-place task, with the potential for further enhancement and broader application in real-world logistics optimization scenarios.