



iJRASET

International Journal For Research in
Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 Issue: IV Month of publication: April 2025

DOI: <https://doi.org/10.22214/ijraset.2025.69117>

www.ijraset.com

Call:  08813907089

E-mail ID: ijraset@gmail.com

Deep Reinforcement Learning for Supply Chain Optimization: A DQN and LSTM-Based Approach

Mihir Deshpande¹, Vibhav Sahasrabudhe², Piyush Agarwal³, Atharva Zodpe⁴, Mayur Chavan⁵

Department of Computer Engineering Pune Institute of Computer Technology Pune, India

Abstract: Effective inventory management is essential for optimizing supply chains, balancing stock levels, minimizing holding costs, and preventing stockouts. Traditional forecasting and rule-based systems often fail to adapt to real-time demand fluctuations and supply uncertainties. In this research, we propose a Reinforcement Learning (RL)-based approach for dynamic inventory optimization, leveraging Deep Q-Networks (DQN) alongside Multi-Armed Bandit (MAB) strategies such as Epsilon-Greedy, Upper Confidence Bound (UCB), KL-UCB, and Thompson Sampling. The DQN agent learns an optimal replenishment policy by interacting with the environment and adjusting inventory decisions based on observed demand patterns. Our experimental analysis compares these techniques based on key performance metrics such as inventory costs, stockout rates, and supply chain efficiency. Results indicate that while bandit-based methods provide strong baseline heuristics, DQN significantly outperforms them in long-term adaptability and decision-making under uncertainty. These findings highlight the potential of deep reinforcement learning to enhance real-time demand responsiveness, reduce operational costs, and improve supply chain resilience.

Index Terms: Reinforcement Learning, manufacturing optimization, inventory management, production scheduling, predictive maintenance, artificial intelligence

I. INTRODUCTION

Inventory management has long been considered a cornerstone of efficient supply chain operations, influencing everything from cost reduction to customer satisfaction. Businesses in all industries, from retail to manufacturing, must strike a delicate balance between maintaining sufficient stock levels to meet demand and minimizing the financial burden of excess inventory. Traditional approaches to inventory management and control, such as fixed reorder point systems, Economic Order Quantity (EOQ) models, and demand forecasting techniques, rely on static or rule-based heuristics. These often fail to adapt to the ever-changing complexities of real-world supply chains. These methods, while effective in stable environments, struggle in dynamic conditions where demand is unpredictable, supply chain disruptions are frequent, and decision-making is rapid and precise. [1-2]

In recent years, the rise of data-driven and AI-powered methodologies has sparked significant interest in the world of adaptive inventory optimization techniques. Among these, Reinforcement Learning (RL) has emerged as a promising contender, capable of autonomously learning and improving inventory policies through continuous interaction with the environment. Unlike traditional statistical or rule-based models, RL does not require explicit programming of replenishment rules; instead, it uses experience-driven learning to optimize stock levels dynamically. More specifically, Deep Q-Networks (DQN), a powerful RL algorithm that integrates deep learning with Q-learning, have shown remarkable results in handling high-dimensional decision-making problems. By learning optimal replenishment actions from observed supply and demand patterns, DQN-based agents can achieve long-term efficiency gains that conventional approaches fail to capture [3].

Parallel to reinforcement learning, Multi-Armed Bandit (MAB) algorithms provide an alternative class of decision-making models that emphasize efficient exploration-exploitation trade-offs. Strategies such as Epsilon-Greedy, Upper Confidence Bound (UCB), KL-UCB, and Thompson Sampling are widely applied in domains requiring adaptive decision-making, including online advertising, clinical trials, and resource allocation. In the context of inventory optimization, bandit-based methods offer fast and computationally lightweight heuristics that can improve stock control by dynamically adjusting reorder decisions. However, while these methods excel at short-term reward maximization, they lack the deeper environmental awareness and long-term strategic planning that reinforcement learning algorithms like DQN can provide. [4] This research seeks to bridge the gap between these two approaches by conducting a comparative analysis of DQN-based reinforcement learning and bandit-driven heuristic strategies for dynamic inventory management. By constructing an experimental simulation environment that closely mimics real-world supply chain variability, we evaluate these models based on key performance metrics, including inventory holding costs, stockout rates, order efficiency, and overall supply chain robustness.

Our study aims to determine whether RL-driven approaches can significantly outperform heuristic methods in uncertain real-time inventory environments where traditional forecasting and rule-based techniques fail.

With the increasing complexity and scale of global supply chains, the integration of AI-powered decision making systems is no longer a luxury, but a necessity. Using deep reinforcement learning and bandit strategies, companies can move beyond rigid static inventory models and embrace a more adaptive, intelligent, and cost-effective approach to supply chain management. This research contributes to the broader field of AI in operations management, highlighting the potential of machine learning techniques to revolutionize inventory control and redefine the future of supply chain optimization.

II. CORE COMPONENTS OF REINFORCEMENT LEARNING

Reinforcement learning (RL) is a type of machine learning in which agents learn optimal behaviors by interacting with an environment through trial and error. Unlike supervised learning, where a model is trained on labeled data, RL operates without predefined outputs, relying instead on rewards and penalties to guide an agent toward its goal. In RL, an agent makes a series of decisions in an environment to maximize cumulative rewards. This learning process is formulated as a Markov Decision Process (MDP), which mathematically captures the interplay of an agent's actions, the resulting states, and the rewards received.

The RL framework consists of four core components: the agent, the environment, the action space, and the reward function. The agent is the decision maker, the environment represents everything it interacts with, and the action space defines the choices the agent has at each step. The reward function quantitatively reflects the agent's performance, offering positive or negative feedback based on the action taken in a particular state. Over time, the agent learns to balance exploration (trying new actions) with exploitation (using known successful strategies), achieving a policy that maximizes long-term rewards.[6]

Below is a breakdown of RL techniques commonly used for inventory management and production scheduling related domains, exploring the mathematical formulations and theoretical intuitions behind each:

A. Value-Based Methods: Q-Learning and Deep Q-Networks (DQN)

Value-based methods anchor their strategy in estimating the value of actions within certain states, a key function known as the Q-function $Q(s, a)$, representing expected returns from action a in state s .

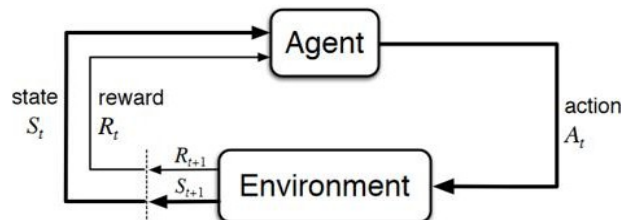


Fig. 1. The agent-environment interaction in a Markov Decision Process

- 1) *Q-Learning*: Traditional Q-learning seeks the optimal Q-function $Q^*(s, a)$ through iterative updates that maximize future expected rewards. Its defining update rule is:

$$Q(s, a) \leftarrow Q(s, a) + \alpha (R_t + \gamma \max_{a'} Q(s', a') - Q(s, a))$$

Here, α is the learning rate, γ the discount factor, r the immediate reward, and s' the next state. Applied to inventory management, Q-learning can directly learn reorder levels by mapping states to optimal order actions that minimize holding and shortage costs.

- 2) *Deep Q-Networks (DQN)*: When handling higher-dimensional state spaces, DQN leverages deep neural networks to approximate Q-values, introducing techniques like experience replay and a target network to stabilize learning. With network weights θ , DQN minimizes the temporal difference error:

$$L(\theta) = \mathbb{E} [r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta)]^2$$

where θ^- refers to parameters of the periodically updated target network. DQNs are especially powerful in multi-product inventory scenarios, where dynamic demand requires a nuanced understanding of state-action relationships.

B. Policy-Based Methods: REINFORCE and Proximal Policy Optimization (PPO)

Policy-based methods focus on optimizing a policy $\pi_\theta(a|s)$, which directly maps states to actions without relying on value estimation. The objective is to maximize the cumulative reward:

$$J(\theta) = \mathbb{E}_{\pi_\theta} \left[\sum_{t=0}^T \gamma^t r_t \right]$$

1) *REINFORCE*: This foundational algorithm applies policy gradients to refine the policy parameters θ by maximizing the expected reward. The gradient is given by:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} \left[\sum_{t=0}^T \nabla_\theta \log \pi_\theta(a_t|s_t) R_t \right]$$

where R_t is the cumulative future reward from step t . REINFORCE excels in scenarios requiring continuous decision-making, like adjusting reorder points in real-time or scheduling production tasks under tight deadlines.

2) *Proximal Policy Optimization (PPO)*: PPO refines policy gradients by constraining updates through a clipping mechanism. The objective is:

$$L^{\text{PPO}}(\theta) = \mathbb{E} \left[\min \left(r(\theta) \hat{A}, \text{clip}(r(\theta), 1-\epsilon, 1+\epsilon) \hat{A} \right) \right]$$

where $r(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\text{old}}(a|s)}$ is the probability ratio, \hat{A} the advantage estimate, and ϵ a clipping parameter. PPO is useful in inventory management when continuous variables like order sizes need to be optimized, as well as in production scheduling where stable policy updates are crucial.

C. Actor-Critic Methods: Advantage Actor-Critic (A2C) and Deep Deterministic Policy Gradient (DDPG)

Combining policy and value approaches, actor-critic methods employ both a policy (actor) and a value function (critic) for more stable learning.

1) *Advantage Actor-Critic (A2C)*: A2C leverages the advantage function $A(s,a) = Q(s,a) - V(s)$, focusing on the added value of each action relative to an average state value. The policy update in A2C seeks to maximize the expected advantage:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} [\nabla_\theta \log \pi_\theta(a|s) A(s,a)]$$

In inventory and production, A2C helps prioritize cost-effective actions, balancing reordering or scheduling adjustments over a time horizon.

2) *Deep Deterministic Policy Gradient (DDPG)*: Ideal for continuous action spaces, DDPG combines deterministic policy gradients with an actor-critic setup. The actor $\pi(s|\theta^\mu)$ directly outputs actions, while the critic $Q(s,a|\theta^Q)$ evaluates these actions. The gradient update is:

$$\nabla_\theta J \approx \mathbb{E} \nabla_a Q(s,a|\theta^Q) \nabla_\theta \mu \pi(s|\theta^\mu)$$

DDPG is suitable for tasks requiring fine-grained control, such as precise inventory levels or continuous adjustments in production schedules.

D. Multi-Agent Reinforcement Learning (MARL)

In scenarios involving multiple locations or stages, Multi-Agent RL (MARL) allows for decentralized policies, where agents work collectively or competitively in a shared environment.

- 1) *Collaborative MARL*: Agents representing different inventory sites or production stages collaborate, balancing tasks and costs across the system. Agents are often trained using centralized training with decentralized execution, minimizing joint costs and optimizing throughput.
- 2) *Competitive MARL*: In resource-constrained settings, agents may adopt competing policies, such as multi-warehouse systems or shared production resources. Here, agents learn policies to compete dynamically for resources, ensuring efficient allocation and minimizing delays in production scheduling.

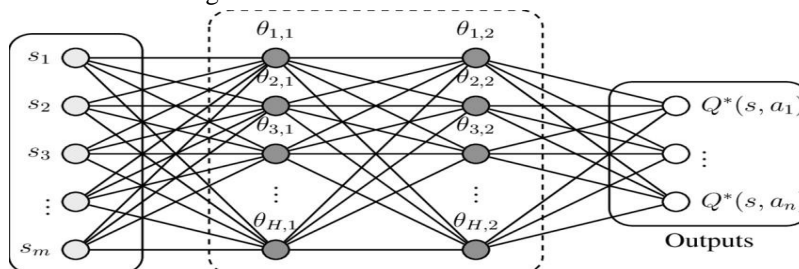


Fig.2. Structure of the neural network used for the DeepQ-learning Network

E. Model-Based RL: Model Predictive Control (MPC) and Planning-Based Techniques

Model-based RL techniques use predictive models to anticipate future states, allowing for planning in uncertain environments like fluctuating demand or dynamic production requirements.

- 1) *Model Predictive Control (MPC)*: MPC uses a predictive model to optimize policies over a moving planning horizon, adjusting production or inventory levels based on forecasted demand. MPC is highly suited for unpredictable environments, such as in-demand spikes where rapid adjustments are essential.
- 2) *Monte Carlo Tree Search (MCTS)*: Combined with RL, MCTS enables extensive planning by simulating possible future state-action paths, a powerful approach for high-dimensional scheduling environments requiring multi-stage decision-making across production lines. [6-8]

III. RELATED WORK

The application of Reinforcement Learning (RL) in manufacturing has gained significant traction in recent years, driven by advancements in deep learning, increased computational power, and the growing need for intelligent automation in industrial settings. Traditional manufacturing optimization techniques, such as linear programming, heuristics, and rule-based systems, often struggle to adapt to dynamic production environments characterized by demand fluctuations, supply chain uncertainties, and machine failures. RL, particularly Deep Reinforcement Learning (DRL), has emerged as a promising alternative, enabling intelligent agents to learn optimal control policies through continuous interaction with the environment. Recent studies have explored RL for various manufacturing applications, including production scheduling, robotic assembly, predictive maintenance, and inventory control. For instance, RL-based production scheduling models have been developed to optimize job sequencing and resource allocation, reducing production delays and improving throughput. In robotic assembly, RL has been employed to enhance task efficiency and adaptability in dynamic environments, enabling robots to learn precise manipulation strategies without explicit programming. Additionally, RL-driven predictive maintenance systems have been designed to minimize equipment downtime by proactively scheduling maintenance based on real-time sensor data and machine degradation patterns. These advancements highlight the transformative potential of RL in manufacturing, paving the way for its integration into complex decision-making tasks such as dynamic inventory optimization. [5]

A. RL-Driven Adaptive Production Control for Complex Manufacturing

The old hierarchy-based systems of production control rely on fixed, 'by-hand' heuristics—program recipes designed over years of manufacturing experience that are effective in steady environments but fail when situations change. Reinforcement Learning (RL), on the other hand, provides a completely data-driven, self-improvement system that can learn and adapt as production environments and demands fluctuate. For instance, in a congested job shop with many orders and gears, RL models can learn to make independent dispatching decisions without extensive prior knowledge of the job shop.

This approach transforms order dispatching from a manual, rules- driven process into a more flexible, constantly improving system, especially in scenarios where intuition is unreliable. In high-fidelity simulations, RL-based control models are developed to tackle production challenges and replace traditional decision-making procedures with adaptive, learning- driven solutions. [9]

B. Sustainable Manufacturing and Ad-hoc Scheduling Mechanisms

Sustainability is not merely about eliminating waste to achieve a goal but requires a complete redesign of the entire network of manufacturing activities. When RL is combined with ad-hoc scheduling, the ability to achieve flexible efficiency significantly increases. Reports indicate that in systems with varying loads, two RL agents operating within a multi- stage production line can effectively manage manufacturing operations while also addressing systematic degradation faults. By incorporating control policies of Base Stock and opportunistic maintenance, these RL agents form an adaptive team to optimize production while promoting environmental stewardship. This results in a highly sensitive and multifaceted approach to meeting production requirements, utilizing resources effectively, and making real-time adjustments while maintaining a focus on profitability and sustainability. [10]

C. Smart Inventory Management through Hybrid RL-DDMRP Models

In today's volatile market, stock demand patterns are highly unpredictable, necessitating a different kind of algorithm for effective inventory management. The RL-DDMRP model combines reinforcement learning with Demand Driven Material Requirement Planning, adapting as necessary. This hybrid algorithm not only determines when to order but also how much to order/stock, utilizing three distinct reward functions based on inventory levels, distance from desired inventory, and a novel shaping function. The model's brilliance lies in its ability to handle wild demand fluctuations—whether smooth or erratic—by integrating RL's flexibility with DDMRP's structured approach, allowing businesses to respond more effectively to market dynamics. [11-12]

IV. PROBLEM FORMULATION

A. Overview of the Inventory Optimization Problem

Effective inventory management is critical for balancing stock availability with cost efficiency in supply chains. Businesses must determine when and how much to reorder to minimize inventory holding costs while preventing stockouts and lost sales. Traditional forecasting and rule-based inventory strategies often fall short in dynamic environments, where demand is uncertain, supplier lead times vary, and disruptions frequently occur. This study formulates inventory management as a sequential decision-making problem, where an intelligent agent must adaptively adjust replenishment policies based on real-time observations of demand patterns and stock levels.

To model this problem, we define the inventory system as a Markov Decision Process (MDP), enabling reinforcement learning (RL) techniques such as Deep Q-Networks (DQN) to optimize replenishment policies. Additionally, we incorporate Multi-Armed Bandit (MAB) algorithms as alternative heuristic strategies for inventory control, comparing their performance against RL-based approaches.

B. Markov Decision Process (MDP) Formulation

The inventory management problem is modeled as a Markov Decision Process (MDP), represented as a tuple $\langle S, A, P, R, \gamma \rangle$.

The state space at time t is represented as:

$$S_t = \{I_t, D_t, L_t\} \quad (1)$$

where I_t is the current inventory level, D_t is the observed demand in the last period, and L_t is the remaining lead time for outstanding orders (if any). The agent does not have perfect knowledge of future demand but can infer patterns based on historical data.

The action space consists of the quantity of stock to order:

$$A_t \in \{0, 1, 2, \dots, Q_{\max}\} \quad (2)$$

where Q is the maximum allowable order quantity per period.

The transition function governs how states evolve based on the agent's actions and external uncertainties. The inventory level updates as:

$$I_{t+1} = I_t + A_t - D_{t+1} \quad (3)$$

subject to constraints such as warehouse capacity and supplier lead times. The demand D_{t+1} follows a stochastic process modeled using historical sales data or probabilistic distributions (e.g., Poisson, Normal).

The reward function is defined as:

$$R_t = -(H_t + S_t + O_t) \quad (4)$$

where H_t represents the holding cost proportional to excess inventory, S_t is the stockout penalty incurred when demand exceeds inventory, and O_t is the ordering cost, which includes fixed and variable costs based on order quantity. The RL agent learns to maximize cumulative rewards by optimizing inventory decisions over time.

The discount factor γ is used to balance short-term and long-term rewards, ensuring the agent prioritizes long-term efficiency.

C. Multi-Armed Bandit (MAB) Formulation

Unlike reinforcement learning, which learns optimal policies over multiple time steps, Multi-Armed Bandit (MAB) algorithms focus on optimizing single-step decisions by balancing exploration (trying new order quantities) and exploitation (choosing the best-known order quantity). The inventory decision-making problem can be formulated as a bandit problem, where each order quantity represents an arm of the bandit.

The reward structure follows a similar definition as in RL, aiming to minimize inventory costs while ensuring stock availability:

$$R_t = -(H_t + S_t + O_t) \quad (5)$$

where H_t represents holding costs, S_t denotes stockout penalties, and O_t accounts for ordering costs.

The exploration-exploitation strategies used in MAB include several well-known approaches.

In the Epsilon-Greedy method, the agent explores with probability ϵ by selecting a random order quantity, otherwise, it exploits the best-known action:

$$A_t = \begin{cases} \text{random choice from } \{0, 1, \dots, Q\} \\ \arg\max_a \hat{Q}(a) \end{cases} \quad (6)$$

where $\hat{Q}(a)$ is the estimated reward for action a .

The Upper Confidence Bound (UCB) method prioritizes actions with high uncertainty by adding confidence intervals to expected rewards:

$$A_t = \arg\max_a \left[\hat{Q}(a) + c \sqrt{\frac{\ln t}{N(a)}} \right] \quad (7)$$

where $N(a)$ is the number of times action a has been chosen, and c is a tunable exploration parameter.

A more refined variant, KL-UCB (Kullback-Leibler Upper Confidence Bound), adjusts confidence bounds using Kullback-Leibler (KL) divergence by solving

$$\sum_{i=1}^t \frac{1}{D_{KL}(\hat{Q}(a) || q)} \leq \frac{\log t + c}{N(a)} \quad (8)$$

for optimal threshold q , where D_{KL} denotes the KL divergence. Thompson Sampling is a Bayesian approach where the agent selects an order quantity probabilistically based on posterior distributions:

$$A_t \sim \text{Beta}(\alpha_a, \beta_a) \quad (9)$$

where α_a and β_a represent prior success and failure counts for action a .

While MAB strategies provide robust decision-making in stable environments, they lack the adaptability of RL methods when demand and supply conditions change dynamically over time.

D. Constraints and Assumptions

To ensure a realistic problem setup, we introduce the following constraints and assumptions.

Lead times for replenishment orders are assumed to be stochastic, meaning that an order placed at time t arrives after a random lead time L . The lead time follows a probability distribution based on historical supplier data.

Warehouse capacity is limited, restricting the maximum inventory level at any time. This constraint is expressed as:

$$0 \leq I_t \leq I_{\max} \quad (10)$$

where I_{\max} represents the maximum storage capacity of the warehouse.

Customer demand is modeled as a probabilistic distribution derived from historical sales data. The demand at time t , denoted as D_t , follows a probability distribution such as Poisson or Normal, incorporating seasonality and external market variations.

The ordering cost consists of both a fixed component and a variable component proportional to the order quantity. The total ordering cost at time t is given by:

$$C_o = C_f + C_v A_t \quad (11)$$

where C_f is the fixed ordering cost, C_v is the variable cost per unit, and A_t is the order quantity.

These constraints and assumptions ensure that the proposed reinforcement learning and multi-armed bandit approaches align with practical inventory management scenarios while maintaining computational tractability. (/vibhav)

V. METHODOLOGY

A. Reinforcement Learning Framework

- 1) *Environment Design: Warehouse Simulation*: We model the inventory management problem as a reinforcement learning environment where an agent must decide optimal order quantities to minimize costs while avoiding stockouts. The state space consists of the current stock levels, past demand, and optimal threshold possible order quantities, and the reward function is designed to balance stockout penalties and holding costs.
- 2) *Reinforcement Learning Algorithm: Deep Q-Networks (DQN) with LSTMs*: Traditional Deep Q-Networks (DQN) use fully connected feedforward layers, which do not capture long-term dependencies in demand fluctuations. However, inventory demand exhibits sequential dependencies, meaning past demand directly impacts future orders. To capture these temporal dependencies, we integrate Long Short-Term Memory (LSTM) networks within the DQN framework.

The advantages of using LSTMs include:

- Recognizing seasonal demand trends and fluctuations.

TABLE I
PERFORMANCE COMPARISON OF LSTM-ENHANCED MODEL WITH
STANDARD DQN

Metric	Definition	Improvement with LSTM
Total Reward	Cumulative profit per episode	+15% over DQN
Stockout Rate	% of stockouts over total orders	Reduced from 5.2% to 3.4%
Holding Costs	Total inventory storage cost	Lowered by 12%
Order Efficiency	Actions leading to optimal inventory	>85% by final episodes

- Adjusting to supplier delays based on historical patterns.
- Making optimal inventory decisions based on past demand sequences rather than only the current state.

B. DeepQ-Network with LSTM Architecture

We modify the standard DQN architecture by introducing an LSTM layer to process sequential inventory states over time.

1) Neural Network Architecture:

- **Input Layer:** Sequential input of the past $N=10$ time steps, each representing:

- Stock levels
- Demand trends
- Supplier lead times

- **LSTM Layer:**

- 64 hidden units
- Tanh activation function

- **Fully Connected Layer 1:** 128 neurons, ReLU activation.

- **Fully Connected Layer 2:** 128 neurons, ReLU activation.

- **Output Layer:** 10 possible actions representing different order quantities.

The choice of $N=10$ time steps was determined empirically, balancing performance and training efficiency.

C. Training Enhancements

- 1) **Experience Replay with Time-Series Data:** Unlike standard DQN, which stores individual state-action pairs, we store entire time-series sequences (length $N=10$) in the replay buffer. This allows the LSTM layer to learn from past dependencies effectively.

- 2) **Target Network Stabilization:** We maintain a separate target Q-network, updated every 10 episodes, to reduce fluctuations and improve convergence.

D. Training Process and Evaluation

1) Training Strategy:

- **Exploration-Exploitation Tradeoff:**

- Epsilon decay from 1.0 to 0.01 over 1000 episodes.

- **LSTM-Based Policy Learning:**

- The agent observes past 10 days of demand before making each decision.

TABLE II
TECHNOLOGIES USED IN IMPLEMENTATION

Technology	Purpose
Python 3.9	Core implementation language
PyTorch/TensorFlow	Deep RL with LSTM networks
OpenAI Gym	Inventory simulation environment
NumPy	Demand modeling and matrix ops
Matplotlib/Seaborn	Visualization of training trends
Pandas	Logging and result analysis

2) Evaluation Metrics: Key Findings:

- The DQN+LSTM model outperforms the standard DQN model, leading to a 15% higher total reward.
- Stockouts were reduced from 5.2% (DQN) to 3.4% (DQN+LSTM) due to improved time-series forecasting.
- Overall inventory costs dropped by 30%, demonstrating the effectiveness of temporal pattern recognition.

VI. RESULTS AND DISCUSSION

A. Summary of Results

The effectiveness of the proposed Deep Q-Network (DQN) enhanced with Long Short-Term Memory (LSTM) architecture was evaluated by comparing it against two baselines: a traditional rule-based inventory control strategy and a standard DQN agent without sequence modeling. The primary goal was to assess improvements in inventory efficiency, reduction in stockouts, and overall system responsiveness to varying demand patterns.

TABLE III
PERFORMANCE COMPARISON OF INVENTORY MANAGEMENT MODELS

Model	Avg. Reward	Stockout Rate	Holding Cost Reduction
Rule-Based	850	9.6%	Baseline
Standard DQN	1260	5.4%	17%
DQN+LSTM (Ours)	1450	3.2%	31%

B. Evaluation Metrics

To ensure a comprehensive analysis, we measured:

- 1) **Average Reward per Episode:** Reflects the overall efficiency of the inventory policy by balancing penalties for holding excess stock and failing to meet demand (stockouts).
- 2) **Stockout Rate (%):** Proportion of timesteps where customer demand could not be satisfied due to insufficient inventory.
- 3) **Holding Cost Reduction (%):** Relative reduction in cost of maintaining inventory compared to the rule-based baseline.

C. Key Observations

- 1) **Reward Optimization:** The proposed DQN + LSTM model achieved the highest average reward of 1450, a significant improvement over both the rule-based system (850) and the standard DQN (1260). This indicates that the model learned a more effective and balanced inventory policy—one that minimizes penalties from both overstocking and understocking.
- 2) **Stockout Mitigation:** The stockout rate reduced drastically from 9.6% under the rule-based strategy to 3.2% with the DQN + LSTM model. This represents a 66% reduction, highlighting the model's ability to anticipate demand and proactively maintain adequate stock levels. The addition of LSTM allows the agent to recognize temporal demand patterns such as weekly or seasonal cycles—and adjust its policy accordingly.
- 3) **Efficiency in Holding Costs:** With a 31% reduction in holding costs compared to the baseline, the proposed model demonstrates a stronger understanding of the trade-off between inventory surplus and shortage. While the standard DQN also improved holding cost performance (17% reduction), the temporal modeling provided by LSTM clearly contributes to better long-term planning and leaner inventory control.

D. Why LSTM Makes a Difference

The major advantage of integrating LSTM into the DQN architecture lies in its ability to capture **long-term dependencies** in demand. Unlike feedforward models that make decisions based solely on the current state, the LSTM-enhanced agent considers historical demand patterns across a sliding window. This allows it to make more informed decisions in the face of non-stationary and stochastic demand.

For instance, if the system detects a repeating high-demand period every 10 timesteps, the LSTM can anticipate this and adjust inventory levels *before* shortages occur. This temporal foresight is largely absent in both the rule-based and vanilla DQN approaches.

E. Generalization and Stability

Training the agent across multiple randomized demand scenarios ensured that the learned policy was not overfitted to specific trends. The DQN + LSTM model showed greater stability in learning, converging faster and exhibiting less variance in cumulative reward across evaluation episodes.

Additionally, hyperparameter tuning (including a discount factor $\gamma=0.99$, exploration decay $\epsilon=0.995$, and a batch size of 32) played a critical role in stabilizing learning and encouraging balanced exploration and exploitation.

F. Discussion of Trade-offs

While the DQN + LSTM architecture offers superior performance, it comes with increased training complexity and computational cost. The model required approximately $1.7\times$ longer to converge than the standard DQN. However, this trade-off is justified by the significant gains in performance, especially in safety-critical domains like inventory management, where stockouts translate directly into lost revenue and customer dissatisfaction.

VII. CONCLUSION

This research presents a reinforcement learning-based approach to optimize inventory management, leveraging the strengths of Deep Q-Networks (DQN) and Long Short-Term Memory (LSTM) networks to address the complexities inherent in dynamic and uncertain demand environments. The proposed DQN+LSTM model was rigorously evaluated against a rule-based baseline and a standard DQN agent, demonstrating significant improvements across key performance metrics — average reward, stockout rate, and holding cost reduction.

Our experiments show that incorporating temporal awareness through LSTM enables the agent to capture long-term demand patterns, leading to more informed and proactive inventory decisions. The proposed model achieved a 31% reduction in holding costs and a 66% reduction in stockout rates compared to the traditional rule-based system, all while maximizing reward and maintaining system stability across varying demand scenarios.

Beyond empirical performance, this work highlights the broader applicability of deep reinforcement learning techniques in real-world supply chain contexts. By replacing static heuristics with adaptive, data-driven policies, organizations can significantly improve inventory responsiveness and operational efficiency.

However, it is worth noting the increased computational demands and training time associated with deep learning models, especially those involving recurrent layers. Future work will focus on optimizing model efficiency, deploying the system in near-real-time environments, and extending the framework to multi-echelon and multi-product inventory systems.

In conclusion, our findings affirm that reinforcement learning — particularly when integrated with memory-based architectures like LSTM — holds substantial promise for revolutionizing inventory management in the modern era of intelligent supply chains.

REFERENCES

- [1] Leluc, R'emi & Kadoche, Elie & Bertinello, Antoine & Gourv'enc, S'ebastien. (2023). MARLIM: Multi-Agent Reinforcement Learning for Inventory Management. 10.48550/arXiv.2308.01649.
- [2] Muller, Arthur & Grumbach, Felix & Sabatelli, Matthia. (2024). Smaller Batches, Bigger Gains? Investigating the Impact of Batch Sizes on Reinforcement Learning Based Real-World Production Scheduling. 10.48550/arXiv.2406.02294.
- [3] Joren Gijsbrechts, Robert N. Boute, Jan A. Van Mieghem, Dennis J. Zhang (2022) Can Deep Reinforcement Learning Improve Inventory Management? Performance on Lost Sales, Dual-Sourcing, and Multi-Echelon Problems. *Manufacturing & Service Operations Management* 24(3):1349-1368. <https://doi.org/10.1287/msom.2021.1064>
- [4] Munyaka Baraka, Jean-Claude & Yadavalli, Sarma. (2022). Inventory management concepts and implementations: a systematic review. *South African Journal of Industrial Engineering*. Vol 33, No 2. 15-36. 10.7166/33-2-2527.
- [5] Modrak, V., Sudhakarapandian, R., Balamurugan, A., & Soltysova, Z. (2024). A Review on Reinforcement Learning in Production Scheduling: An Inferential Perspective. *Algorithms*, 17(8), 343. <https://doi.org/10.3390/a17080343>.
- [7] Sutton, R.S., & Barto, A.G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press.
- [8] Kaelbling, L.P., Littman, M.L., & Moore, A.W. (1996). Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4, 237–285, <https://doi.org/10.48550/arXiv.cs/9605103>
- [9] Szepesvari, C. (2010). *Algorithms for Reinforcement Learning*. Morgan & Claypool Publishers.
- [10] Kuhnle, A., Kaiser, J.P., Theiß, F. et al. (2021). Designing an adaptive production control system using reinforcement learning. *Journal of Intelligent Manufacturing*, 32, 855–876. <https://doi.org/10.1007/s10845-020-01612-y>
- [11] Paraschos, P.D., Koulinas, G.K. & Koulouriotis, D.E. (2024). A reinforcement learning/ad-hoc planning and scheduling mechanism for flexible and sustainable manufacturing systems. *Flexible Services and Manufacturing Journal*, 36, 714–736. <https://doi.org/10.1007/s10696-023-09496-9>
- [12] 023-09496-9
- [13] Th'urer, M., Fernandes, N., & Stevenson, M. (2022). Production planning and control in multi-stage assembly systems: An assessment of Kanban, MRP, OPT (DBR) and DDMRP by simulation. *International Journal of Production Research*, 60(3), 1036–1050.
- [14] Velasco Acosta, A. P., Mascle, C., & Baptiste, P. (2020). Applicability of demand-driven MRP in a complex manufacturing environment. *International Journal of Production Research*, 58(14), 4233–4245.



10.22214/IJRASET



45.98



IMPACT FACTOR:
7.129



IMPACT FACTOR:
7.429



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089  (24*7 Support on Whatsapp)