Optimizing Explainable Deep Q-Learning via SHAP, LIME, & Policy Visualization

By

Tesfahun Yemisrach Wendimu

Submitted to the School of Information Technology and Engineering

In partial fulfillment of the requirements for the degree of

Master of Science in Artificial Intelligence

Supervised by: Dr. Beakal Gizachew

, Co-advisor: Dr. Natnael Argaw

School of Information Technology and Engineering

College of Technology and Built Environment

Addis Ababa University

Addis Ababa, Ethiopia

June , 2025

APPROVAL

This is to certify that this thesis titled **"Optimizing Explainable Deep Q-Learning via SHAP, LIME, & Policy Visualization"** is prepared by Tesfahun Yemisrach Wendimu and submitted in partial fulfillment of the thesis-option requirements for the Degree of Master of Science in Artificial Intelligence at School of Information Technology & Engineering, Addis Ababa Institute of Technology.

Name	**Signature**	**Date**
Dr. Beakal Gizachew		
_____	_____	_____
(Advisor)		
Dr. Mesfin Abebe		
_____	_____	_____
(External Examiner)		
Dr. Fantahun Bogale		
_____	_____	_____
(Internal Examiner)		
Dr. Henok Mulugeta		
_____	_____	_____
(Chairperson)		

ABSTRACT

Reinforcement learning (RL) has demonstrated remarkable promise in sequential decision-making tasks; however, its interpretability issues continue to be a hindrance in high-stakes domains that demand regulatory compliance, transparency, and trust. Post-hoc explainability has been investigated in recent research using techniques like SHAP and LIME; however, these methods are frequently isolated from the training process and lack cross-domain evaluation. In order to fill this gap, we propose an explainable Deep Q-Learning (DQL) framework that incorporates explanation-aligned reward shaping and model-agnostic explanation techniques into the agent's learning pipeline. The framework exhibits broad applicability as it is tested in both financial settings and traditional control environments. According to experimental findings, the explainable agent continuously performs better than the baseline in terms of explanation fidelity, average reward, and convergence speed. In CartPole, the agent obtained a LIME fidelity score of 87.2% versus 63.5% and an average reward of 190 versus 130 for the baseline. It produced an 89.10% win ratio, a Sharpe Ratio of 0.4782, and a return of 154.32% in the financial domain. The development of transparent and reliable reinforcement learning systems is aided by these results, which demonstrate that incorporating explainability into RL enhances interpretability as well as stability and performance across domains.

Key words: Deep Q-Learning, Explainable AI, SHAP, LIME, Reinforcement Learning, Policy Visualization

Acknowledgements

"Wisdom comes from the Lord; understanding is from Him." — Proverbs 2:6

I want to sincerely thank the following people for their invaluable assistance in finishing this thesis:

I am most grateful to the Almighty God for providing me with endurance, strength, and wisdom during this journey. The essential turning point in my education and individual growth occurred because of His unmerited favor. During challenging moments of uncertainty along with exhaustion I found solace and inspiration through His presence. I present this work as a dedication to Him since I feel immense gratitude for each grace that has entered my life.

I am deeply grateful to my advisor, Dr. Beakal Gizachew, for his expert knowledge and steadfast support which helped me develop this work. Through his leadership I learned to analyze deeply while developing strong work habits that led me to academic achievement. I deeply value the time and dedication that Dr. Beakal Gizachew provided to help me handle every stage of this research project.

My heartfelt thanks go to all my friends for their unwavering support throughout this journey. I want to thank Mr. Chala Bekabil in particular, whose encouragement, companionship, and wise counsel were a continual source of inspiration. Throughout this process his presence provided both strength and meaning through our conversations as well as his emotional backing.

My treasured family holds an unbreakable debt from my heart. My academic journey began with their unyielding love and their many sacrifices and prayers which continue to drive me forward to this day. Your belief in my potential gave me the courage to seek out my goals with determination. Their heartfelt support both spiritually and emotionally formed the foundation for this achievement to become reality.

Lastly, I would like to express my sincere gratitude to everyone who helped me finish this thesis, whether it was through encouragement, technical support, or moral support. I pray that God will bless you all abundantly and that the results of this work will be a testament to the strength of knowledge, faith, and perseverance.

Dedication

To my family.

. . .

To my academic advisor.

. . .

To my best friends.

. . .

Table of Contents

Abbreviaions

Symbol	Meaning
AI	Artificial Intelligence
AMDP	Aggregated Markov Decision Process
ANN	Artificial Neural Network
ARMA	Auto-Regressive Moving Average
CNN	Convolutional Neural Network
DANSMP	Dual Attention Networks for Stock Movement Prediction
DDQN	Double Deep Q-Network
DL	Deep Learning
DNN	Deep Neural Network
DQL	Deep Q Learning
DQN	Deep Q Network
DRL	Deep Reinforcement Learning
DP-LSTM	Differential Privacy-Inspired Long Short-Term Memory
HFT	High Frequency Trading
IRR	Investment Return Rate
KNN	K-Nearest Neighbors
LIME	Local Interpretable Model-Agnostic Explanations
LSTM	Long Short-Term Memory
MacroHFT	Memory-Augmented Context-Aware Reinforcement Learning on High-Frequency Trading
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MARL	Multi Agent Reinforcement Learning
MDP	Markov Decision Process
MKG	Market Knowledge Graph
ML	Machine Learning
MLP	Multilayer Perceptrons
NIFTY	National Stock Exchange Fifty
NLP	Natural Language Processing
OHLCV	Open-High-Low-Close-Volume Price Data
PG	Policy Gradient
POMDP	Partially Observed Markov Decision Process

PPO	Proximal Policy Optimization
PRE	Prediction Rule Ensembles
RL	Reinforcement Learning
RMSE	Root Mean Squared Error
ROI	Return on Investment
SA	Sentiment Analysis
SAMDP	Semi Aggregated Markov Decision Process
SCM	Structural Causal Models
SDM	Sequential Decision Making
SHAP	Shapley Additive Explanations
SMDP	Semi Markov Decision Process
SMP	Stock Market Prediction
SMP	Stock Market Prediction
SOFNN	Self-Organizing Fuzzy Neural Network
SOTA	State of the Art
SPP	Stock Price Prediction
SR	Sharpe Ratio
ST	Stock Trading
STS	Stock Trading System
SVM	Support Vector Machine
TD3	Twin Delayed Deep Deterministic Policy Gradient
t-SNE	t-distributed Stochastic Neighbor Embedding
XAI	Explainable Artificial Intelligence
XRL	Explainble Reinforcement Learning
VADER	Valence Aware Dictionary for Sentiment Reasoning
VMSE	Value Mean Square Error

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

1.1 Background

In recent years, the growing complexity of real-world decision-making environments driven by dynamic system behaviors, uncertainty, and high-dimensional data has increased the demand for intelligent agents capable of optimizing outcomes. These environments, whether in industrial control systems, robotics, healthcare scheduling, or financial decision-making, present a common challenge: how to model and automate sequential decision-making under uncertainty.

The risks and perils of overfitting in machine learning are well-established. However, most treatments for this, including diagnostic tools and remedies, were developed for supervised learning cases [1]. With the advent of artificial intelligence (AI) and machine learning (ML), particularly reinforcement learning (RL), there has been a surge in interest in applying these methods to problems that require sequential decision-making. However, traditional ML models often struggle to capture long-term dependencies in decision-making processes. As observed by Mehtabetal., "deep learning–based models have much higher capability in extracting and learning the features of time-series data than their corresponding machine-learning counterparts" [2]. Reinforcement learning has demonstrated promise in optimizing decision-making processes in uncertain and dynamic environments that necessitate sequential decision making. Reinforcement learning involves teaching an agent to make decisions by interacting with an environment and receiving feedback. Among the RL algorithms, DQL stands out because of its ability to scale to high-dimensional state spaces using deep neural networks to approximate action-value functions. DRL integrates the perception of DNN with the decision-making ability of reinforcement learning to achieve the goals of simulating the cognition and learning mode of humans [3]. This makes DQL a strong candidate for addressing decision-making problems in which traditional tabular methods are insufficient.

While the success of DQL in domains such as game playing and control is well documented, its application in real-world systems still faces major hurdles. One critical issue is the lack of transparency in decision making. Compared to the burst of XAI research in supervised learning, explainability in model-free reinforcement learning has hardly been explored [4]. As noted by Arrieta et al., "explainability is one of the main barriers AI is facing nowadays in regards to its practical implementation" [5]. Since deep RL models are frequently regarded as "black boxes," it can be challenging to decipher the motivations behind specific actions. Due to the requirement for regulatory compliance, user trust, and accountability, this lack of interpretability may impede adoption in delicate industries like healthcare or finance.

Furthermore, standard DQL models frequently ignore problems with overfitting, risk awareness, and generalizability to unknown environments in favor of performance metrics like reward maximization. These issues are especially pertinent to partially observable or non-stationary systems, where models need to adjust to changing circumstances. Additionally, learning stable and reliable policies is made more difficult by reliance on delayed and sparse feedback.

Enhancing DQL architectures with transparency tools like SHAP and LIME is obviously necessary as the need for decision-making systems that are both explainable and performant increases. Developers and users can better comprehend and validate the learned policies with the aid of these tools, which can offer insights into an agent's internal decision logic.

This research attempts to create a DQL framework that strikes a balance between decision quality, transparency, and adaptability. It is situated at the nexus of explainable AI (XAI) and deep reinforcement learning. This study aims to enhance the performance and reliability of DQL agents across various sequential decision-making domains by directly integrating explainability into the learning and assessment procedures.

1.2 Motivation of the Study

The increasing emphasis on explainability in artificial intelligence (AI) and the growing need for intelligent systems capable of making sequential decisions in complex, uncertain, and dynamic environments are the driving forces behind this study. The ability to understand and trust the decisions made by machine learning models has become just as crucial as the models' performance as artificial intelligence (AI) continues to be used in delicate fields like healthcare, finance, autonomous systems, and industrial control.

Many real-world tasks, from automated trading and resource allocation to robotics and medical treatment planning, require agents to learn from interactions and to optimize long-term outcomes rather than immediate gains. RL, particularly DQL, has proven effective for such tasks because of its capacity to handle sequential decision-making with delayed feedback and high-dimensional input spaces.

But despite its popularity, DQL has a significant drawback: because of its intricacy, it is frequently opaque and challenging to understand. Stakeholders in high-stakes situations need to know why an agent makes a specific choice, particularly when the results could have an impact on safety, ethics, or finances. Interpretability, explainability, and transparency are key issues in introducing Artificial Intelligence methods in many critical domains [6]. Deep reinforcement learning's lack of transparency has become a hindrance to its wider use, underscoring the requirement for models that are not only precise and self-governing but also explicable and responsible.

The opportunity to confrontthis challenge was the motivation for this study. In this work, we try to bridge thedivide between explainability and performance by extending the DQL framework to integrate explainability strategies such as LIME and SHAP. The goal is to build a non-myopic agent (i.e. an agentthat learns to make wise choices for which its underlying algorithms and rationale can be easily explained), since this is important to make users trust our agent or reason about how ethically it can be used, when used in restricted setting.

Furthermore, this study is especially pertinent to the Ethiopian context. With the anticipated opening of the Ethiopian Stock Market, there is an increasing demand for transparent AI systems that support financial analysis and decision-making. This study offers a foundational AI framework that strikes a balance between accuracy, transparency, and usability, and while its insights and tools are generalized beyond financial markets, they can be modified to support Ethiopia's developing financial infrastructure.

This research establishes a robust DQL framework which researchers can apply across multiple domains to advance explainable reinforcement learning standards. The study creates practical solutions together with academic support while constructing guidelines for ethical AI implementation across national and global domains.

1.3 Statement of the Problem

Deep Q-Learning continues to advance as a successful sequential decision-making method but multiple challenges block its adoption in crucial real-world industries such as healthcare and finance and robotics. Key issues include a lack of interpretability, poor generalization, sensitivity to dynamic environments, and weak evaluation protocols for explainability [4, 5, 7]. Traditional DQL agents may perform poorly or behave erratically in these environments because they frequently have non-stationary dynamics, sparse feedback, and evolving constraints.

One central problem is the absence of robust off-policy reinforcement learning algorithms for decomposed rewards that are both sound and convergent [8, 9]. Traditional DQL algorithms limit the creation of modular policies because these algorithms cannot handle detailed behavioral control and reward attribution requirements. Conventional DQL algorithms which learn through stable simulations face major challenges when they need to function in noisy real-world systems that have dynamic conditions.

The second major issue is the explainability of DQL agents. While these models can learn effective policies, they largely function as black boxes [10], making it difficult to interpret or justify decisions. Adoption in safety-critical applications, where regulatory compliance, human oversight, and accountability are necessary, is hampered by this lack of transparency. Intrinsically interpretable models often sacrifice performance, while post-hoc methods like SHAP and LIME can provide model-agnostic insights without modifying the architecture [11]. Nevertheless, there is still little incorporation of these explanations into the training procedure.

Generalization also poses a challenge. Many DQL agents perform well in narrow, controlled environments but fail to transfer their learning to slightly altered conditions, reflecting overfitting to simulation biases [12]. This problem is compounded by the cost and complexity of designing realistic simulation environments that match real-world distributions. Additionally, the lack of integrated risk-aware strategies leaves agents vulnerable to unstable high-reward policies that are suboptimal under uncertainty.

Real-time responsiveness and computational efficiency form crucial requirements in financial trading and robotics applications. DQL models operate within exact latency boundaries yet they need to preserve both robustness and clear decision-making processes. Current systems struggle to meet these demands while offering interpretable outputs [13].

Finally, the evaluation of explainability in DQL remains immature. Existing literature focuses on reward-based metrics while neglecting fidelity, consistency, and user-centered explanation quality [14]. Reinforcement learning requires new validation systems which test both performance metrics together with interpretability aspects. The establishment of reliable evaluation methods becomes critical for developing reinforcement learning agents which deliver precise results alongside transparent and trustworthy decision-making capabilities in practical non-laboratory situations.

1.4 Research Questions

This research study aims to find answers to a series of questions which emerge from the problems and limitations that appear when applying DQL in sequential decision-making fields.

1. In what ways can DQL be improved to handle overfitting, volatility, and uncertainty in various sequential decision-making contexts?
2. How can explainability strategies (like SHAP and LIME) be successfully incorporated into DQL without sacrificing real-time decision-making or learning performance?
3. Which measurement methods can be applied to assess the quality of reinforcement learning agents explanations and their predictive behavior?

1.5 Objective of the Research

1.5.1 General Objective

The overall goal of this study is to improve model explainability and optimize DQL for sequential decision-making in a variety of contexts without sacrificing policy efficacy or predictive accuracy.

1.5.2 Specific Objectives

The specific objectives of this study are as follows.

1. To create a DQL framework that incorporates explainability tools like SHAP and LIME to enable clear, intelligible decision-making.
2. To develop a reward function that explicitly promotes explainability through explainability-guided penalties, balancing transparency with learning effectiveness.

3. To evaluate the quality of post-hoc explanations in different dimensions –accuracy, consistency, and feature relevance in diverse domains like finance and control.

4. To examine the effect of explainability on generalization, policy behavior and user trust in dynamic scenarios.

5. To employ holistic measures across both performance and explainability using state-of-the-art explainability and RL measures.

1.6 Contribution of the Study

Our research addresses the main drawbacks of conventional DQL models, especially with regard to transparency, generalization, and real-time performance in sequential decision-making tasks, and thus significantly advances the developing field of reinforcement learning.

The enhancement of explainability and predictive accuracy is one of the main contributions. The research combines post-hoc explainability tools such as SHAP and LIME with DQL framework while traditional DQL models function as black-box systems. The resulting interpretable agent enhances human trust and understanding by providing transparent explanations which matter most during high-risk scenarios and real-time applications.

The development of a distinctive reward structure which matches domain semantics represents a major advancement. Through this approach the agent strengthens its decision-making capabilities by facing penalties when it produces solutions that lack coherence. The combination of explainability with rewards establishes an innovative framework for value-based reinforcement learning systems.

Furthermore, by confirming the suggested method's generalizability across several domains, this study advances the field. Tests were carried out in benchmark control environments (such as CartPole and Acrobot) and real-world-like settings, showing that the suggested explainable DQL framework can preserve explainability and performance outside of its original domain. The framework's potential as a general tool for explainable sequential decision-making is supported by this cross-domain validation.

This study advances Explainable AI best practices through new quantitative metrics which evaluate explanation quality based on fidelity alongside feature importance stability and local interpretability scores. The evaluation metrics establish a comprehensive baseline for analyzing

XAI in reinforcement learning since this field typically lacks standard evaluation procedures.

The model underwent final optimization to achieve real-time operation capabilities which ensures its scalability and responsiveness in changing environments. The framework operates efficiently in real-time applications including autonomous systems together with industrial automation and robotics and adaptive control environments.

This research expands existing academic understanding together with practical artificial intelligence implementations by solving problems related to explainability and generalization along with evaluation methods and real-time decision-making challenges. The study presents a reinforcement learning framework that researchers and practitioners can apply to different sequential decision tasks through scalable interpretable efficient methods.

1.7 Scope/Delimitation of the Study

1.7.1 Scope of the Study

This research establishes a DQL framework which unifies model interpretability with prediction accuracy for running generic sequential decisionmaking problems. The study examines both practical and technological elements needed to develop a scalable and transparent and trustworthy reinforcement learning system.

The core of this study is the application of DQL to decision-making tasks in environments where agents must learn optimal policies through trial and error. Rather than being confined to a single domain, such as finance, the framework was validated across multiple environments with different dynamics and reward structures, such as CartPole and Acrobot. These standards were used as stand-ins to evaluate how broadly applicable the suggested strategy was.

This research employs post-hoc explainability techniques SHAP and LIME to demonstrate the internal decision-making structure of DQL agents. The framework analyzes the impact of different input features on action selection to enhance the interpretability of learning processes for real-time and safety-critical applications.

Adding explainability to the reward system is another crucial area of attention. In order to promote behaviors that are both optimal and interpretable, the framework incorporates a modified reward function that penalizes actions that lack coherence or alignment.

The research examines scalability together with real-time decision capabilities. The agent

operates effectively in environments which require quick decisions that apply to multiple situations.

A wide range of performance and explainability metrics were employed to assess the model's efficacy. These include quantitative explainability metrics like the fidelity of feature attributions and common reinforcement learning indicators like cumulative reward and learning stability. This guarantees a comprehensive evaluation of the framework from both functional and interpretive viewpoints.

1.7.2 Delimitations of the Study

In order to preserve the viability and clarity of the scope, a number of boundaries have been established, even though the goal of this study is to show the generality and explainability of DQL across various sequential decision-making tasks.

- Selective Domain Coverage: Both the financial and non-financial domains have validated the framework. Experiments were specifically carried out on standard control environments like CartPole and Acrobot, as well as stock market trading (with sentiment-enhanced environments). However, this study excluded more general real-world domains like industrial control systems, robotics, healthcare, and autonomous driving.

- Limited Financial Market Scope: The experiments are limited to stock price movement prediction using sentiment signals and historical market data within the financial domain. Cross-market interactions, portfolio-level modeling, and other financial instruments (like options and futures) are outside the purview of this study.

- Simplified Environment and Reward Structures: The reward functions together with their explainability alignment modifications rest upon particular domain assumptions and all testing occurs in simulated environments. Future research needs to extend reward-shaping techniques to support both complex multi-agent systems and real-time environments.

- Emphasis on Post-Hoc Explainability: This study pursues the combination of SHAP and LIME as post-hoc explainability tools for integration. The analysis avoids domain-specific explainability methods which need model adjustments or self-explanatory policy frameworks.

1.8 Structure of the Document

The remainder of this paper is organized as follows. The next chapter reviews the existing literature on reinforcement learning in sequential decision-making contexts, with particular attention to DQL, explainable AI techniques such as SHAP and LIME, and prior applications in both financial and control domains. This section also identifies the key limitations of the current methods that this study aims to overcome.

Subsequently, the methodology chapter introduces the proposed Explainable DQL framework. It details the agent architecture, data processing pipeline, sentiment-aware reward shaping, and integration of post hoc explainability tools. It also describes the design of experiments across multiple domains, including a financial trading environment and control tasks, such as CartPole and Acrobot.

The results section presents a comprehensive evaluation of the model performance. It compares baseline and explainable variants in each domain using both traditional performance metrics (e.g., ROI and Sharpe ratio) and explainability evaluations (e.g., SHAP consistency and LIME fidelity). This section includes reward progression curves, visualizations, and a cross-domain analysis of generality.

The discussion section follows, reflecting on the key insights derived from the experiments, trade-offs between explainability and performance, and practical implications of deploying XRL systems across domains.

Finally, the conclusion summarizes the contributions of the study and offers recommendations for future research directions, including extensions to more complex environments and real-time deployment scenarios.

Chapter 2

Literature review

2.1 Background

Sequential decision-making problems arise in domains where an agent must take a series of actions over time to maximize long-term rewards, often under conditions of uncertainty. These problems are commonly modeled using frameworks such as the Markov Decision Process (MDP) and span a wide range of applications, including robotics, healthcare, operations management, autonomous systems, and finance. Previous successful attempts of model-free and fully machine-learning schemes to the algorithmic trading problem, without predicting future prices, are treating the problem as a Reinforcement Learning (RL) one [15]. In recent years, Reinforcement Learning (RL) has emerged as a powerful approach for solving such problems, enabling agents to learn optimal policies through interaction with the environment and feedback in the form of delayed rewards.

Among the family of RL algorithms, DRL, which integrates deep neural networks with traditional RL methods, has achieved significant breakthroughs in tackling complex, high-dimensional problems. Notably, DQL has been successfully applied to domains such as game playing, robotic control, and financial decision-making. DQL leverages a deep neural network to approximate the Q-value function, facilitating policy learning in environments in which the state and action spaces are large or continuous. Its ability to learn directly from raw data without handcrafted features makes it an appealing choice for dynamic decision-making.

Despite these advances, classical DRL models often behave as black boxes, rendering their internal decision logic opaque to users and stakeholders. This lack of transparency poses a serious challenge in real-world applications, especially in high-stakes domains such as healthcare, automated trading, and safety-critical control, where trust, interpretability, and regulatory compliance are crucial. Consequently, recent research has increasingly focused on Explainable

Reinforcement Learning (XRL), which seeks to make RL agents more transparent and accountable. Techniques such as SHAP and LIME have shown promise in interpreting both the global and local decision patterns of complex models; however, their integration into the RL training loop remains limited in most studies.

The financial market, particularly stock trend prediction, serves as a prime example of a real-world environment that embodies the challenges and opportunities of explainable sequential decision-making. Stock markets are known for their high volatility, nonlinear behavior, and dependence on a wide range of quantitative and qualitative factors, including macroeconomic indicators, investor sentiment, and breaking news. Traditional trading models often rely on handcrafted technical indicators, which may fail to generalize across market conditions. Rather than exploiting predefined handcrafted features, can we learn more robust feature representations directly from the data? [16]. Yes, we can, and this reflects a growing trend in financial machine learning that emphasizes the use of deep learning for automated and adaptive feature extraction. Recent studies on financial AI have demonstrated that when properly optimized and regularized, DRL agents can effectively automate portfolio allocation and risk-sensitive decision-making. However, these models often lack transparency, which remains a barrier to their practical adoption in regulated financial ecosystems.

This literature review examines the key methodologies, architectures, and challenges in the application of RL, particularly DQL, to sequential decision-making tasks. It explores the evolution of model design strategies, including advancements in performance, stability, and generalization capabilities. Special attention is given to the emerging field of explainable reinforcement learning, where the goal is to enhance the transparency and trust in RL-based decision systems. By critically analyzing a wide spectrum of prior works across domains such as finance, control systems, and autonomous agents, this section aims to highlight the state of the art, identify persistent limitations, and outline how the proposed research builds upon and contributes to the existing literature.

2.2 Reinforcement Learning for Sequential Decision-Making

RL algorithms have long been recognized as powerful tools for optimal sequential decision making. The framework is concerned with a decision maker, the agent, that learns how to behave

in an unknown environment by making decisions and observing their associated outcomes. Through repeated experience, the RL agent seeks to infer an optimal decision-making policy, or a set of action rules that would yield the highest, usually long-term, expected utility. Today, a wide range of domains, from economics to education and healthcare, have embraced the use of RL to address specific problems [17].

Reinforcement learning problems involve learning what to do and how to map situations to actionsso as to maximize a numerical reward signal. In an essential way They are closed-loop problems because the learning system's actions influence its later inputs. Moreover, the learner is not told which actions to take, as in many forms of machine learning, but instead must discover which actions yield the most reward by trying them out [18].

A Markov Decision Process (MDP), represented as a tuple M = (S, A, P, R,), is typically used to model an environment in reinforcement learning (RL). In this model, S stands for the state space, A for the action space, P for the transition probabilities, R for the reward function, and is the discount factor. The agent interacts with the environment in discrete time steps, observes state st, acts at rt, and gets feedback in the form of rewards rt to direct the learning process.

Reinforcement learning has proven effective across a wide range of industries, including robotics, gaming, resource management, and finance. The ability of RL to independently learn policies in dynamic and unpredictable environments without supervision is its main strength. This is why RL is particularly attractive for real-world decision-making situations where self-learning, flexibility, and long-term planning are essential.

Many RL algorithms have been created over time, ranging from more sophisticated actor-critic and policy-based frameworks to more straightforward value-based techniques like Qlearning and SARSA. DQL is one of the most popular RL approaches in complex environments because of its ability to scale to high-dimensional spaces through deep neural networks.

Much of the work in this thesis is motivated by the fact that, despite their success, conventional RL approaches still have a number of drawbacks, especially with regard to stability, exploration-exploitation balance, and interpretability issues.

2.3 Deep Q-Learning for Sequential Decision Making

Reinforcement learning in high dimensional and continuous state spaces is made possible by DQL, an extension of the traditional Q-learning algorithm that incorporates deep neural networks to approximate the Q-value function Q(s,a). DQL, which was first introduced by Mnih et al., has shown impressive performance in tasks like playing Atari games straight from raw pixel inputs where conventional tabular methods are impractical.

DQL maintains a value function Q(s,a;), parameterized by a deep neural network, which estimates the expected cumulative reward of taking an action a in a given state s. Instead of using a Q-table, DQL trains a neural network to estimate the Q-values, which act as a function approximator. This way the neural network with parameters serves as an approximate of the optimal function: Q(s, a;) Q(s, a) [19].The training process minimizes the loss between the predicted and target Q-values derived from the Bellman equation. To stabilize learning, DQL employs several enhancements, such as experience replay (storing and sampling past transitions to break temporal correlations) and target networks (holding fixed parameters to calculate target Q-values over multiple updates).

One of DQL's notable benefits is its scalability. The use of deep neural networks enables the algorithm to operate effectively in large and continuous state spaces that are otherwise unmanageable with traditional tabular Q-learning techniques. By enabling agents to learn from their errors without the aid of expert demonstrations or supervised labels, DQL also promotes autonomous learning. Its general-purpose framework also makes DQL applicable to a wide range of domains, including robotics, gaming, and finance.

However, DQL has some drawbacks. However, it suffers from overestimation and instability, particularly when dealing with stochastic dynamics or sparse rewards. To converge to a stable policy, DQL often requires large computational resources and extensive interactions with the environment, in addition to being sample inefficient. The incapacity to be interpreted, especially in practical situations, is a more serious issue. Because DQL's deep networks are "black boxes," it is challenging to comprehend or justify the agent's choices. Trust in automated systems may be damaged by this lack of transparency, especially in fields where choices have significant ethical or financial ramifications. Another major drawback is DQL's poor generalization of DQL. Although the model may perform well within the training environment, it frequently fails

to adapt when faced with new or slightly altered scenarios, resulting in performance degradation.

These limitations, especially the issues of explainability and generalization, highlight the need for improved DQL frameworks. In high-stakes applications such as healthcare, autonomous systems, and financial decision-making, addressing these gaps is essential for both the functional reliability and responsible deployment of reinforcement learning models.

2.4 Explainability in Reinforcement Learning

The success of Deep RL could augur an imminent arrival in the industrial world. However, like many Machine Learning algorithms, RL algorithms suffer from a lack of explainability. This defect can be highly crippling for many promising RL applications (defense, finance, medicine, etc.) [20]. Reinforcement learning (RL) often yields robust but opaque models with difficult-to-understand decision-making processes, particularly when deep neural networks are used. Due to their black-box nature, RL systems raise concerns regarding their security, accountability, and dependability—particularly in high-stakes scenarios. Explainability is now crucial for RL agents to meet regulatory requirements and give human operators insights into agent behavior as they are used more and more in industries like autonomous driving, finance, and healthcare.

Intrinsically interpretable models and post-hoc explainability techniques are the two main strategies for pursuing explainability in RL. Intrinsically interpretable approaches, which design the RL algorithm or its architecture to be understandable by default, include replacing deep neural networks with decision trees or linear models. Nevertheless, this frequently results in a decrease in expressiveness and performance, which restricts their use in intricate settings.

On the other hand, post-hoc techniques seek to elucidate the choices made by a black-box model that has already been trained. The model-agnostic nature of techniques like SHAP and LIME has led to their increasing popularity.SHAP values quantify the contribution of each input feature to a decision and offer a unified measure of feature importance based on cooperative game theory. In contrast, LIME builds a local surrogate model, often linear, around a specific prediction to approximate the decision boundary of the black-box model in that region. Both methods have been successfully applied in classification and regression settings and, more recently, in reinforcement learning to explain value functions, action selections, or policy shifts.

Post-hoc methods are useful, but they have problems when used in reinforcement learning

(RL). In contrast to supervised learning, reinforcement learning (RL) incorporates nonstationary policies, delayed rewards, and sequential decisions, all of which make it more difficult to interpret a single action. Furthermore, more thorough interpretive frameworks are required because explanations are frequently sought not only at the action level but also at the trajectory or policy level.

Recent work has begun to bridge this gap by adapting SHAP and LIME for value-based RL models. These initiatives include using reward decomposition to allocate rewards to latent goals or sub-policies, visualizing action-value surfaces, and explaining Q-values over time. Additionally, visualization tools like saliency maps and policy heatmaps have been developed to offer intuitive views of how agents respond to changes in their environment. However, there are still a lack of established benchmarks and assessment metrics for explainability in RL.

Explainable Reinforcement Learning (XRL) has emerged as a research subfield in response to the increasing demand for transparent decision-making in reinforcement learning. The goal is to create RL agents that make decisions that are both optimal and human-understandable. This involves ensuring the consistency and fidelity of explanations, developing metrics for evaluating the quality of explanations, and designing user-centered interfaces that effectively communicate agent intentions. More responsible and reliable RL systems for study and application might be possible as this field develops.

2.5 Application Domains

2.5.1 Control Domains

Reinforcement learning has been extensively tested in standard control environments to evaluate its generalizability and robustness. Platforms like OpenAI Gym offer benchmark environments for evaluating how well reinforcement learning agents perform in limited but interpretable scenarios, such as CartPole, MountainCar, and Acrobot.We constructed simulated physical environments with varying levels of difficulty to test our algorithm. This includes classic reinforcement learning environments such as cartpole [...] using both a low-dimensional state description (such as joint angles and positions) and high-dimensional renderings of the environment [21].

In the CartPole environment, the agent's task is to balance a pole on a moving cart by applying forces to the left or right. The CartPole environment offers a useful testbed for assessing learning

stability and reward optimization in spite of its simplicity. The Acrobot environment is perfect for studying policy learning in underactuated systems because it presents a more difficult task, requiring the agent to use torque-based control to swing a two-link pendulum to a predetermined height.

This study extended the Explainable DQL framework to the CartPole and Acrobot domains. To maintain consistency, the neural architecture and hyperparameters utilized is same with the financial domain too, and reward-shaping techniques were modified to penalize actions that were inconsistent or misaligned. These tests made it possible to thoroughly examine whether SHAP and LIME, the model's explainability mechanisms, could generalize to other domains and decision-making situations.

The explainability-enhanced DQL and the baseline DQL performed noticeably differently. Although they had somewhat lower peak rewards, the explainabilityaware models in CartPole showed more stable policies and smoother learning curves. However, Acrobot presented significant challenges for both agents; the explainable agent fared worse, highlighting the necessity of carefully balancing domain-specific dynamics with explainability constraints.

To ensure robustness, transparency, and policy clarity across contexts, these experiments highlight the importance of validating reinforcement learning models in both general-purpose control tasks and specialized domains, such as finance.

2.5.2 Financial Domain

Partially Observed Markov Decision Process (POMDP)

The Partially Observed Markov Decision Process (POMDP) is an extension of the Markov Decision Process (MDP) that models decision-making problems in which the agent has limited or "partial" visibility into the true state of the system. The stock trading problem is being modeled as a Partially Observed Markov Decision Process (POMDP), which can be formulated by describing its State Space, Action Space, and Reward Function [22]. The authors framed the trading problem as a POMDP, which allows the agent to operate under certainty and make sequential decisions using time-dependent financial data. They added a twin delayed deep deterministic policy gradient (TD3) algorithm to the model in order to reduce overestimation bias and enhance trading performance in a continuous action space.This continuous space enables

the model to make dynamic trading decisions (buy, sell, or hold) across a range of assets.Their agent state representation includes portfolio information, sentiment analysis scores from news headlines, and technical indicators. The model's 2.68 Sharpe Ratio in back testing demonstrated how well it performed in relation to market volatility. Notably, the model's forecasting ability was improved by adding sentiment analysis and technical indicators, which increased profitability. The model's responsiveness was enhanced by the continuous action space of the TD3 algorithm, outperforming more conventional models with discrete action spaces. Given its success in trading automation, DRL may surpass current machine learning techniques in financial market forecasting and portfolio management.

This study has limitations even though it shows a comprehensive approach to trading automation.The model's applicability in intricate trading environments may be limited by its simplification of real-world elements like market impact, slippage, and liquidity constraints. Next, in addition to transaction costs, it is better to consider other potential financial risks and taxes in the model. Another point that can be raised at this time is the limitation of the backtesting dataset, which consists of a small selection of assets and historical data; hence, the use of more diverse data could provide a more comprehensive assessment of the model's robustness.

Future research should explore the scalability of the model to handle a wider array of assets and markets (such as commodities or currencies). In addition, explainability is also a research gap for this study, as understanding the decision-making rationale is essential for real-world adoption in financial contexts. In addition, the model was tested only on historical data; thus, adapting it for real-time trading environments, where market conditions are constantly evolving, would be a valuable extension.

Bi-typed Hybrid-Relation Market Knowledge Graph (MKG)

A knowledge representation technique designed to capture intricate relationships and interconnected entities within financial markets is the bi-type hybrid relationship market knowledge graph (MKG). A bi-typed hybrid relational market knowledge graph that simulates the intricate relationship between businesses and their executives is used in this study to investigate stock movement prediction.The authors developed a dual attention network for stock movement prediction (DANSMP) model, which leverages both explicit and implicit relations within the MKG. First, the stock sequential embedding is learned with historical price and media news

via multimodal feature fusion and sequential learning. (II) Second, a dual attention network is proposed to learn the stock relational embedding based on the constructed MKG. (III) Finally, the combinations of sequential and relational embeddings are utilized to make stock predictions [23]. The model outperforms state-of-the-art baselines and enhances stock trend forecasting by incorporating historical stock prices, media sentiment, and executive relationships. Two recently developed datasets, CSI100E and CSI300E, are used to test the model's efficacy. According to the results, the model performs better than other models in terms of profitability metrics like the Sharpe ratio (SR) and investment return rate (IRR), which show lower risk and higher returns in simulated trading, as well as prediction accuracy.

This study's main advantages are its thorough relational modeling, which combines explicit and implicit relationships to increase the accuracy of stock movement predictions. Additionally, the dual attention mechanism facilitates a sophisticated comprehension of interconnected entities, improving the model's predictive and financial performance.

This study has some limitations. One of these is that the model was tested only on two specific datasets (CS1100E and CS1300E). This may restrict the generalizability of the findings to other markets. The other is that although the MKG is effective for the datasets used in this study, scalability to larger and more complex data could be challenging. The research has some gaps, including testing cross-market applicability, real-time trading adaptability, and incorporating an interpretability component that helps increase model trustworthiness.

Prediction Rule Ensembles (PRE)

PRE is a statistical learning method that combines the benefits of rule-based models and ensemble learning approaches to increase predictive accuracy and interpretability. The hybrid prediction model is the most popular method for stock price prediction, producing superior results [24]. The researchers created a novel hybrid model to predict stock price trends by combining the benefits of PRE and DNN. The technique uses a DNN model with hidden, input, and output layers in addition to PRE to generate decision trees that produce binary prediction rules based on the root mean square error. The PRE was fed the stock uptrend data, generating various tree combinations. Each prediction rule is evaluated based on logical statements [24]. One of the main conclusions of this study is that the hybrid model performs better than the DNN and ANN models alone, improving RMSE by 5–7%. This illustrates how well decision

rules and deep learning methods work together. It finds that using multiple moving averages is particularly helpful for identifying stock uptrends, and it also adjusts DNN hyperparameters to improve prediction accuracy.

The hybrid model's strong predictions, increased accuracy over individual models, and thorough assessment using accepted metrics like MAE and RMSE are some advantages of this strategy. However, the study has some limitations, such as relying solely on a few technical indicators (moving averages) and testing the model on Indian stock data, which may restrict its applicability to other markets.

However, this study has some limitations. One is the lack of reinforcement learning techniques that could enhance the predictive power by enabling the model to learn from trading actions. Another is the neglect of sentiment analysis and macroeconomic variables, which may provide more profound understanding of changes in stock prices. Another flaw in the model is its emphasis on predicting a single stock; expanding its use to include multiple assets for portfolio management could give investors more useful information.

Deep Q-Network (DQN) with Policy Gradient

DQN is essentially a value-based method that approximates Qvalues using a deep neural network. This enables an agent to determine the optimal course of action in a particular state based on anticipated future rewards. The model can directly learn a policy and use gradients to optimize it by incorporating the policy gradient method. Changing the parameters to maximize the reward function is the primary method of putting the policy gradient (PG) strategy into practice. The secret to this strategy is adjusting the policy parameters through extensive computation and iteration. The adjusted parameters will be included in the policy-making equation and modified continuously to accommodate more accurate policies [25]. By employing sophisticated DRL techniques, the authors hope to improve prediction accuracy and adaptability to swift market changes while addressing issues like stock price volatility and nonlinearity. As the DRL agent engages with a simulated market environment, the methodology models stock prediction as a Markov decision process (MDP). The PG method modifies the model parameters to optimize the expected rewards, while the DQN method uses a CNN combined with Q-learning to learn the best course of action based on historical data.

This approach's advantages include sophisticated DRL techniques that efficiently handle volatility and adjust to market swings, as well as an understandable evaluation procedure that contrasts the DQN and PG approaches. The study's shortcomings, however, are its small dataset, reliance on daily stock data, and single-agent model, which might not adequately represent intricate market dynamics. Moreover, the model excludes sentiment analysis and macroeconomic factors, which could produce more detailed forecasts.

This study's research gaps include using higher frequency data for HFT applications, integrating external economic and sentiment data for enriched predictions, and investigating multi-agent systems for more realistic simulations.

Deep Reinforcement Learning (DRL)

Deep Reinforcement Learning is a branch of machine learning combines reinforcement learning (RL) with deep learning techniques. It involves training agents to make sequential decisions by interacting with an environment in which they learn optimal behaviors through trial and error, guided by rewards or penalties. The highly complex nonlinearity of a DNN can approximately simulate and describe the complexity of the influencing factors, as deep nonlinear topologies can be built to represent complex high-dimensional functions by stacking the hidden layers of nonlinear activation functions. In addition, they are numeric, data driven, and adaptive to analyze inaccurate and noisy data and have been extensively used to predict time series [26]. This study investigates the potential of DRL to predict stock market movements and enhance trading strategies. It offers a framework for teaching an autonomous agent to make sequential trading decisions, like buying, holding, or selling stocks, by interacting with a simulated stock market environment. In order to guarantee meaningful input representations during model training, feature engineering techniques were employed. The state space was defined by the authors using stock-related indicators, the action space by standard trading actions (buy, sell, and hold), and the reward function by minimizing risk exposure and maximizing profit.

The study's strong points include an adaptive learning methodology, a comprehensive comparison of different DRL models, and efficient risk management strategies. Interestingly, the Double DQN model demonstrated its resilience by providing lower volatility and more consistent returns.

The requirement for extensive processing power and careful hyperparameter tuning complicates the deployment of these models. Another limitation is that the study primarily relies on historical data, limiting its applicability in real-time trading scenarios. In addition, one major area that can be seen as a gap is real-time trading implementation, which would move these models from theory to live market deployment, and another is the lack of explainability and interpretability, as reinforcement learning models are often black boxes.

Deep Reinforcement Learning with Sentiment Analysis

Reinforcement learning is a branch of machine learning in which an agent learns to make decisions by interacting with its environment. Sentiment Analysis identifies whether the expressed sentiment is positive, negative or neutral. The stock market has a huge amount of data and many influencing factors, and most of them are retail investors who do not have enough expertise to obtain useful information directly from it, which makes it difficult to benefit [27]. The methodology for assigning sentiment scores to financial texts includes sentiment analysis using natural language processing (NLP) techniques.These sentiment scores are then used as input features alongside stock price data. To increase accuracy, the model uses feature engineering to combine sentiment data with past price movements. Backtesting the model against machine learning baselines and conventional trading strategies allows us to assess its performance.

Compared to reinforcement learning models that solely consider stock price, the proposed model outperformed them. Prediction accuracy is increased by incorporating sentiment analysis, particularly in volatile times. The model also achieves strong riskadjusted returns, as indicated by the Sharpe ratio metrics, and demonstrates how combining textual and numerical data leads to more adaptive trading strategies.The strength of this study is the integration of multiple data sources, which boosts the predictive accuracy of the model. The reinforcement learning agent consistently enhances its trading approach, resulting in consistent performance improvements.The model's reliance on sentiment data , which may introduce bias from exaggerated or misleading information in financial texts, is one possible flaw.It also needs a lot of training resources and is computationally demanding.

Deep Q-Learning

Researchers sometimes mash up classic Q-Learning with modern deep neural networks under the catchy name deep Q-learning. In that setup a bulky network stands in for the tidy action-value table, churning out an estimate of how much cash or candy a single move is likely to rack up, provided you keep playing smart after the first nudge. Accurate price prediction is a difficult undertaking for analysts and investors alike because the stock market is a dynamic, complex system influenced by a wide range of factors [28]. Deep neural networks and Q-learning principles are combined in DQL, a subset of reinforcement learning, to allow agents to discover optimal strategies in complicated situations by trial and error [28]. This study emphasizes the growing role of artificial intelligence and machine learning in financial markets and aims to develop a predictive model that improves trading efficiency and forecasting accuracy. The authors implemented a reinforcement learning framework in which Q-learning was used to make sequential trading decisions. A deep neural net now stands in for the old Q-table, letting the agent chew on states and actions until it finally stumbles on a strategy that squeezes out every last reward. During the tuning runs, we kept a close eye on Mean Squared Error, accuracy of the forecasts, and the bottom-line profit to see whether the shiny architecture was learning anything useful.

The study's main conclusions demonstrate that the Deep Q-Learning model is capable of successfully spotting lucrative stock trading opportunities. This shows dynamic adaptability to market fluctuations, which helps investors make better decisions. DQL optimizes trading strategies and demonstrates exceptional predictive accuracy.

One obvious upside is the heavy lift from advanced AI-tools. A suite of cutting-edge algorithms, paired with an adaptive learning hook, lets the model refine its strategy on the fly as new market data rolls in. A battery of performance gauges, from Sharpe ratios to drawdown figures-backs the claim that this system could really give traders an edge.

Every piece of research has its rough edges, and this one is no exception. Were pulling insights from a handful of datasets, so the results may not travel well beyond the companies we sampled. Training a DQL engine demands heavy computational firepower, which not every lab can muster on a tight budget. Plus, we've sidelined big players such as news sentiment and key macro indicators that can swing stock prices on short notice.

Deep Learning with Statistical Models

Deep learning sits squarely at the intersection of big data and advanced mathematics, letting layers of digital neurons pull out patterns most of us would miss by eye. In finance, traders feed these models everything from three years of price ticks to the morning headlines, and the algorithms then spit out probability clouds showing where a stock might drift next. Predicting a fast and accurate model for stock price forecasting is a challenging task and an active area of research, where the best way to forecast stock prices is yet to be found [29].The stock market is the backbone of any economy and it defines the profit maximization and the minimization in risk [29]. This paper puts deep learning side-by-side with old-school statistical techniques to see which one can better guess tomorrow's stock price. A pretty big question, really, because traders live and die by that one number. the research runs everything from trusty ARIMA and good-enough linear regression all the way to finicky LSTM and CNN stacks. The goal is to lay out, in plain sight, how well each method handles the markets messy, twisting, jittery nature.

As part of the methodology, the model is trained and evaluated using historical stock prices, technical indicators, and macroeconomic variables as input features. Traditional models and deep learning approaches were evaluated, and R-squared, MAE, and RMSE scores were used to gauge performance.

The study's main conclusion is that deep learning models—particularly LSTMs—perform better than conventional techniques at identifying intricate nonlinear patterns and long-term dependencies. In short-term projections, ARIMA does fairly well, but it has trouble with longer-term patterns. According to the authors, improved predictive accuracy could result from a hybrid strategy that combines the advantages of statistical modeling and deep learning.

This study's benefits include its thorough and extensive model comparison, use of several performance metrics for impartial assessment, and usefulness for investors and financial analysts looking for reliable forecasting tools. This study's sound methodology makes it an invaluable resource for comprehending the strengths and weaknesses of various modeling techniques.

Nevertheless, a number of shortcomings have been identified. Trust and adoption in the financial industry are hampered by the deep learning models' black-box nature and lack of transparency regarding prediction-making. Additionally, the study overlooks external variables like investor sentiment and macroeconomic developments that are essential to actual financial

decision-making. Inappropriate use of reinforcement learning, which can improve enhanced prediction quality through a feedback-driven approach, is one of the gaps that can be mentioned. The study also lacks explainability techniques, which are crucial for understanding model behavior, especially in high-stakes financial settings.

Deep RL with Cascaded LSTM Network

A sophisticated deep learning architecture known as a cascaded LSTM network uses a series of Long Short-Term Memory (LSTM) layers stacked one after the other, with the output of one LSTM layer acting as the input for the subsequent one. This hierarchical structure allows the model to capture complex temporal dependencies and multilevel patterns in sequential data more effectively than a single layer LSTM. Financial market data are filled with noise and are unstable, and also contain the interaction of many unmeasurable factors [30]. The reinforcement learning approach learns how to maximize the objective function, which can be achieved by maximizing the value function here, during the training phase [30]. The proposed methodology involves a DRL agent interacting with a simulated stock market environment and learning trading strategies through reward-based feedback. The temporal patterns in stock price movements were successfully captured by the feature extraction process using a cascaded LSTM network. The Sharpe ratio, cumulative returns, and maximum drawdown are among the performance metrics used to assess the system through back testing on actual stock market data.

The main conclusions show that, in terms of profitability and risk-adjusted returns, the model performs better than conventional machine learning-based trading strategies. Trading decisions become better informed when cascaded LSTM is used because it improves the model's capacity to extract valuable time-series features. The DRL framework offers strong support for real-time automated trading by allowing agents to dynamically adjust to shifting market conditions.

One of the study's advantages is the efficient modeling of sequential dependencies in stock prices using LSTM networks, along with the DRL framework's proven versatility. The model's practical utility gains credibility from empirical validation on real-world data, and the results demonstrate notable gains in stability and profit over baseline models.

Nonetheless, a number of shortcomings were identified. First, deployment in resource-constrained

environments may be limited by the high computational demands resulting from the combined use of DRL and cascaded LSTM networks. Additionally, the model is not explainable, which is a crucial prerequisite for financial applications where decision-making transparency is essential.

The study stops short of testing whether the model holds up in different markets and under shifting economic skies. That leaves a big question mark over its real-world portability. Explainability tools are left on the shelf, so readers have no simple way to see inside the models black box. The research also skips a look at DQL, a leaner flavor of reinforcement learning that might do the same job with less fuss. Outside signals-sentiment waves, macroeconomic pulses-are not woven into the framework, and that gap could cost the model some forecasting edge.

2.5.3 Other Application Areas

Reinforcement Learning (RL) has demonstrated significant potential across a diverse array of domains beyond financial trading and control systems. This breadth underscores the versatility and adaptability of RL methodologies in addressing complex real-world challenges.

In robotics, RL algorithms such as PPO and DQN have been pivotal in enabling robots to acquire skills such as locomotion, manipulation, and navigation. These algorithms facilitate learning from high-dimensional sensory inputs, allowing robots to perform tasks with a level of autonomy and adaptability that was previously unattainable. As noted by Schulman et al., PPO has demonstrated strong performance on high-dimensional continuous control problems involving a 3D humanoid, where "the robot must run, steer, and get up off the ground, possibly while being pelted by cubes."[31].

The healthcare sector has also benefited from the application of RL. Notably, RL has been employed to optimize treatment strategies, personalize medication dosing, and manage chronic disease. For instance, RL frameworks have been developed to adjust insulin dosing for patients with diabetes, aiming to maintain optimal blood glucose levels while minimizing the risk of hypoglycemia. As noted by Yu et al., "RL approaches have attracted increasingly high attention in personalized, patient specific glucose regulation in AP systems." [32].

In traffic signal control, RL has been instrumental in developing adaptive systems that respond to real-time traffic conditions. MARL approaches have been utilized to coordinate multiple

traffic signals, thereby reducing congestion and improving traffic flow efficiency. These systems model each traffic signal as an agent that learns optimal policies through interactions with the environment and neighboring agents. As noted by Zhang et al., "Capturing the spatial-temporal correlation among traffic lights under the framework of Multi-Agent Reinforcement Learning (MARL) is a promising solution." [33].

Natural Language Processing (NLP) has seen the integration of RL, particularly through Reinforcement Learning from Human Feedback (RLHF). This approach has been used to fine-tune language models and enhance their ability to generate coherent and contextually appropriate responses. RLHF has been a cornerstone in the development of advanced conversational agents that align model outputs more closely with human preferences.

2.6 Related Work

A study entitled "Graying the Black Box: Understanding DQNs" introduced by Zahavy, Baram, and Mannor [7], offers a unique framework designed to make Deep Q-Networks (DQNs), which are usually regarded as opaque or "black box" models, easier to understand. In order to make policy analysis easier, the authors suggest a technique called the Semi Aggregated Markov Decision Process (SAMDP), which combines temporal and spatial abstraction. The study examines DQN behavior in a variety of domains, such as Gridworld environments and Atari games, by applying clustering techniques and dimensionality reduction methods like t-SNE. By identifying important abstractions and sub-goals that the agent learned, the authors were able to demonstrate that DQNs naturally create hierarchical representations during training, which contributes to the explanation of their empirical efficacy.

This study's methodology combined empirical evaluation, abstraction modeling, and manual state analysis. In order to visualize neural activations from the last hidden layer of the DQN, manual state clustering was first carried out using t-SNE. Saliency maps and handcrafted features were then applied to interpret policy structures and state dynamics. After that, the authors presented a brand-new abstraction model called the SAMDP, which combines spatial abstraction from Aggregated MDPs (AMDPs) with temporal abstraction from SMDPs. High-level patterns and policy behavior can be extracted thanks to this model's facilitation of spatiotemporal clustering and the inference of skills and transitions. VMSE, inertia, entropy, and

intensity factor are some of the evaluation criteria used in this study to evaluate the quality and dependability of the abstractions. Additionally, a correlation analysis between greedy policies and reward trajectories is used to validate policy consistency. The framework's applicability in revealing interpretable structures within DQN policies was demonstrated through testing across a variety of empirical domains, including Gridworld, Breakout, Seaquest, and Pacman.

The study's main conclusions show that DQNs naturally create spatiotemporal abstractions, proving that the models can pick up internal hierarchical policy structures without direct training. The authors were able to restore interpretable sub-goals and behavioral skills through the suggested SAMDP framework, providing a useful tool for comprehending and visualizing policy dynamics. In addition to revealing significant structural patterns, the visual analysis of neural activations and state trajectories also reveals the advantages and disadvantages of DQN behavior. This study also emphasizes the model's potential in shared autonomy scenarios, where predictions of policy degradation can effectively guide human intervention, improving safety and confidence in reinforcement learning systems.

This study has a number of limitations in spite of these contributions. The approach's inability to scale to high-dimensional or continuous action spaces, which are frequently encountered in real-world applications, stems primarily from its reliance on manual feature engineering and visualization techniques. The whole method leans on policies that are, for all practical purposes, almost set in stone. That rigidity can crack when the environment tosses unexpected surprises our way. Reliability, in other words, takes a hit once real randomness steps in. On top of that, researchers have mostly tested the framework inside the safe, predictable bubble of lab simulations-s think Atari scoreboards and the like. That narrow proving ground leaves a pretty wide gap when you start asking whether the system will hold up in high-stakes situations where people actually bet their lives or livelihoods.These restrictions draw attention to significant research gaps. First, there is still work to be done to generalize the SAMDP and related visualizations to real-world tasks. Second, this study highlights the need for automated abstraction discovery since manual state and skill identification hinders scalability and real-world deployment. Finally, by integrating the SAMDP framework into reinforcement learning training pipelines directly rather than only post-hoc, learning efficiency and policy transparency could be significantly improved.

A study that entitled as A Robust Predictive Model for Stock Price Prediction Using Deep Learning Natural Language Processing that conducted by Sidra Mehtab Jaydip Sen [34]. presents a comprehensive framework for predicting stock price movements by integrating machine learning, deep learning, and sentiment analysis. The authors employed a diverse set of classification and regression models, including Logistic Regression, KNN,ANN, and SVM, as well as LSTM for deep learning in addition to sentiment analysis. The key findings of this study indicate that the random forest achieved the best classification accuracy for predicting NIFTY 50 stock price movements, surpassing other models. In regression tasks, LSTM performed with the highest accuracy compared to traditional models like decision trees and random forests because it can handle sequential data and identify temporal patterns.Additionally, with low MAPE and high predictive accuracy, the SOFNN model with sentiment data enrichment outperformed the LSTM model.

This study's hybrid approach, which combines deep learning and machine learning models with sentiment analysis to increase predictive power, is one of its strongest points. Additionally, the model's robustness is highlighted by its use of cross-validation and multiple model comparisons, which provide a comprehensive assessment of different approaches. The study's narrow focus on NIFTY 50 stock price movements, however, limits its generalizability to other markets. Furthermore, the model's scope excludes other macroeconomic variables like interest rates and geopolitical events that could improve the predictive accuracy.

Another research that conducted which is related to financial domain is entitled as Application of Deep Reinforcement Learning in Stock Trading Strategies and Stock Forecasting, introduced by Yuming Li et al., [35]. It investigates how Deep Reinforcement Learning (DRL) models—more especially, Deep Q-Networks (DQN), Double Deep Q-Networks (DDQN), and Dueling DQN—can improve decision-making in intricate financial contexts when applied to stock trading and prediction strategies. DRL models use neural networks to evaluate past data in order to forecast stock prices and produce successful trading plans.

One of the study's main conclusions is that, in terms of profitability in the stock market environment, DQN performs better than its improved counterparts, DDQN and Dueling DDQN, which was unexpected. Despite their general benefits in applications such as gaming, the fact that DDQN and Dueling DDQN do not perform better than DQN in this financial context

indicates that the unique dynamics of financial markets render these model variations incompatible. There are certain limitations in contrast to their main conclusions. First, the study's generalizability is limited by the dataset's scope, which is restricted to 10 stocks and may not accurately reflect the larger market. Furthermore, the model's performance during the training and testing stages might have been impacted by data inconsistencies, such as the different durations of historical data for every stock.

A research entitled Explainable Reinforcement Learning Through a Causal Lens, introduced by Madumal et al. [36],presents a novel XRL framework that generates counterfactual and contrastive arguments for the decisions made by model-free reinforcement learning agents using SCMs. The purpose of this study is to train agents to answer "why" and "why not" questions by teaching them the causal relationships between environmental variables. The authors assessed the framework empirically using the real-time strategy game StarCraft II and computationally across six reinforcement learning environments. The outcomes show that the causal explanation model has the potential to be a human-aligned method of enhancing transparency and trust in reinforcement learning systems by improving users' task prediction accuracy as well as their satisfaction and comprehension.

The Action Influence Model is a novel framework that extends SCMs by explicitly incorporating actions and their causal effects on environmental variables. This framework is used in the methodology proposed in this study to integrate SCMs with reinforcement learning. The authors created a mechanism for causal explanation generation that uses learned causal chains to produce both complete and minimally complete explanations in order to produce insights that are understandable to humans. By comparing real decisions with hypothetical alternatives, this method also makes it possible to create contrastive explanations. Regressors like linear regression, decision trees, and MLPs are used to approximate causal functions during training through experience replay in the framework's structural equation learning process. For empirical validation, the framework was computationally evaluated across six reinforcement learning domains, including environments such as Cartpole and StarCraft II, using task prediction accuracy as a measure of explanation fidelity. In addition, a human-subject study involving 120 participants was conducted, in which the participants observed RL agents, received explanations, and were assessed based on their task prediction performance, perceived explanation quality using Likert scales, and trust in the agent's decision-making process.

According to the study, causal explanations outperformed the baseline explanation techniques in terms of task prediction accuracy and user satisfaction. Moreover, contrastive explanations enhance users' comprehension of agents' decision-making processes by simulating "what if" situations. The mean task prediction score for users who received causal explanations was 10.9 out of 16, while the score for users who did not receive explanations was 8.5. Despite not always being statistically significant, the increases in trust scores for causal explanations showed a consistent upward trend. The creation of a human-aligned explanation model based on social and cognitive psychology, along with its encouragement of counterfactual reasoning, which allows for insightful reflection on agent behavior, are the study's strong points. In order to show that learning structural causal models is feasible even in model-free reinforcement learning settings, the authors also put into practice a strong experimental framework that integrated computational evaluation with user studies.

This strategy has a number of significant drawbacks despite its benefits. Its flexibility and scalability are limited by its reliance on predefined causal graphs, especially in high-dimensional or dynamic environments. Furthermore, the approach is not appropriate for continuous control tasks like the BipedalWalker because it is restricted to discrete action spaces and small state environments. The quality of the approximated causal models, which might not be adequate in more complex settings, also has a significant impact on the fidelity of the explanations. These restrictions point to important areas that require more study. Most importantly, in order to facilitate wider real-world applicability, the field needs techniques for the automated discovery of causal structures. Extending these methods to reinforcement learning agents that function in continuous action spaces is another unresolved issue. Furthermore, post hoc explanations are the main focus of the current study; real-time, interactive explanations during agent deployment should be investigated in future studies. Lastly, research into customized explanation techniques that take into consideration each user's preferences and epistemic state is urgently needed.

A study entitled DP-LSTM: Differential Privacy-Inspired LSTM for stock prediction using financial news was presented by Li et al. [37]. Introduced the DP-LSTM model that uses a deep learning approach designed to enhance stock price prediction by integrating sentiment analysis of financial news with stock price data. It combines the ARMA model with an LSTM neural network. Sentiment scores from financial news were derived using the VADER sentiment analysis tool, and these scores, along with historical stock data, were used to predict SP 500 stock

prices.

They used the ARMA model, which is classified as one of the key methodologies used for stock price prediction, with a sentiment-ARMA extension that integrates sentiment data for enhanced accuracy. One of the strengths of the model is its innovative incorporation of sentiment analysis, which captures sentiment and news impact on stock trends, and the differential privacy mechanism, which helps prevent reliance on biased information.

The models training and testing data span only for a short period of time which could limit its generalizability to longer-term stock predictions is one of the weakness of the study. The model was also validated only on the S&P 500 index, which raises questions regarding its adaptability to other financial indices. Furthermore, sentiment analysis was limited to four major news outlets, restricting the diversity of sentiment data and potentially increasing bias from these sources.

A study entitled "Explainable Reinforcement Learning: A Survey", introduced by Erika Puiutta et al. [38], gives a thorough rundown of the techniques created to improve the transparency of models used in reinforcement learning. The authors provided a thorough taxonomy of XRL techniques in order to address the growing concern about the opacity of high-performing RL algorithms. These methods are categorized based on their scope (global vs. local) and timing (intrinsic vs. post-hoc). Programmatic policies, causal models, mimic learning, and hierarchical explanations are just a few of the representative approaches that are examined in the survey, with a focus on how each approach contributes to interpretability. This study also emphasizes the value of interdisciplinary contributions, especially from cognitive science and human-computer interaction, to better match explanation strategies with user expectations and needs. By establishing the groundwork for future XRL models that not only function well but also promote trust and understanding in practical deployments, this study makes a substantial contribution to the field.

This study uses a clear classification framework to arrange and assess current XRL techniques. The authors divided these approaches into two main categories: timing and scope. They differentiate between local explanations, which concentrate on elucidating the specific actions or decisions made by the agent, and global explanations, which seek to make the entire decision-making policy understandable. In terms of timing, the taxonomy distinguishes between post-

hoc methods, which obtain explanations after training using surrogate models, rule extraction, or visualization techniques, and intrinsic methods, where interpretability is directly incorporated into the model architecture. Using this framework, the authors provide a structured lens through which to view the current state of explainability in reinforcement learning by methodically reviewing 13 representative XRL approaches, such as causal explanation frameworks, Linear Model U-Trees (LMUTs), hierarchical skill acquisition models, and Programmatically Interpretable Reinforcement Learning (PIRL)..

The survey found a number of significant trends influencing the XRL field. It should be mentioned that due in large part to the intrinsic complexity of reinforcement learning systems, post-hoc explanation techniques currently rule the field. The authors do point out a significant drawback, though: a lot of current methods ignore the human viewpoint, resulting in explanations that might be sound technically but unclear or difficult for end users to understand. Contrastive explanations, which respond to queries such as "Why this action instead of that?" stood out among the other tactics as being especially successful in enhancing user comprehension. Additionally, this study supports the consistent use of "interpretability" as a general term and draws attention to the absence of standardized terminology in the field. Importantly, this highlights how challenging it is to strike a balance between model performance and interpretability. One of the main benefits of this study is its comprehensive taxonomy, which provides an orderly framework for evaluating XRL techniques.The survey is both educational and useful because the writers provide critical evaluations of each method's drawbacks and applicability. The paper huddles around a simple question: how do we coax computers into giving honest, down-to-earth explanations of their decisions? It answers by leaning hard on human-centered design. Designers, psychologists, A.I. researchers, and HCI specialists are invited to crowd the same brainstorming table. A shared roadmap emerges, one that prizes reliability, clarity, and that stubborn quality people keep asking for-accuracy.

Every research effort has its blind spots, and this survey is no exception. Readers hoping for fresh data sets or side-by-side performance numbers will walk away disappointed because the paper mostly offers a big-picture overview rather than hands-on experiments. Another worry is that most of the techniques it describes are built around tidy symbolic actions or discrete choices. Those same methods might struggle once theyre dropped into the messy, high-dimensional worlds we see outside the lab. Without widely accepted benchmarks for explainable

reinforcement learning, pinning down which approach is truly better remains a frustrating guessing game. This study identifies a number of unexplored research avenues to build on these limitations. Notably, context-aware explanation models are required, which are able to modify explanations according to a user's task familiarity, cognitive preferences, or prior knowledge. Additionally, the field has not advanced real-time interactive explanation mechanisms that work dynamically during agent operation instead of after the fact. Furthermore, little is known about XRL in multi-agent systems and applied fields like stock trading and finance. In order to objectively evaluate the impact and efficacy of explanations across various use cases, the authors conclude by advocating for the development of formalized evaluation metrics, such as fidelity, human satisfaction, and trust.

A research entitled Stock Market Prediction Using Reinforcement Learning with Sentiment Analysis which introduce by Xuemei Li Hua Ming [39]. presented an innovative DQL model that integrates sentiment analysis for trend prediction in the stock market. The authors prioritize predicting upward or downward movements rather than exact stock prices, using sentiment data from news headlines as additional input for informed trading decisions. They used a model called VADER to assess sentiment from news articles and a technical indicator to increase the accuracy of trend forecasts. The model demonstrated significant performance improvements over a traditional DQL model that only used technical data.

With a Sharpe ratio of 3.65 (as opposed to 1.6 in the traditional model), the DQS model's portfolio value is 83% higher than that of the baseline DQ model, demonstrating its ability to identify profitable trades even in volatile situations like the COVID-19 pandemic. The study does have certain limitations, though, such as testing only one stock data set, which limits generalizability.

The last related study that we want to see is MacroHFT: Memory Augmented Context-Aware Reinforcement Learning on High Frequency Trading, conducted by Zong et al. [40]. This paper introduces MacroHFT, a sophisticated reinforcement learning (RL) framework created especially for cryptocurrency markets' high-frequency trading (HFT). The framework addresses two prevalent RL problems in HFT: overfitting and adapting to volatile market conditions. MacroHFT has a structure for hierarchical RL, exercising specialized subagents constituting a hyperagent. In other words, while each sub-agent is independently trained to behave best in

particular market conditions (volatility and trend), the hyper-agent combines these sub-agents into a meta-policy that works together coherently. The first steps of MacroHFT techniques are decomposition of the market and classification of the data by volatility and trend so that every sub-agent may be trained for unique circumstances.

One of the study's main conclusions is that across all tested cryptocurrency markets, MacroHFT outperformed the baseline models in terms of profitability, achieving higher total returns and risk-adjusted profits. It is appropriate for the particular requirements of high-frequency trading due to its flexibility in responding to a variety of market conditions, which guarantees consistent performance in both trending and volatile environments. One of this study's key advantages is the memory mechanism that allows the model to react swiftly to abrupt changes in the market.

The main limitation of this study is that MacroHFT's scalability may be limited when dealing with larger datasets or real-time applications due to the multi-agent architecture and complex memory module, which significantly increase computational complexity. Additionally, the model's data sources are restricted to price and order book data; it does not take into account outside variables that might improve its decision-making, such as news sentiment or macroeconomic influences.

Table 2.1: *SOTA Comparison in Explainable and Risk-Aware Deep Q-Learning Frameworks*

Papers	**DQL**	**Explainability**	**Generalization**	**Risk-Aware**	**Domain-Specific**	**Real-Time**	**Evaluation Metrics**
[14]	✓	✓	Partial	✓	✗	Partial	✓
[39]	✓	✗	✗	✓	✓	✗	✓
[7]	✓	✓	✗	✗	✗	✗	✓
[38]	✓	✓	✗	Partial	✗	✗	✓
[35]	✓	✗	✓	✗	✓	✗	✓
[25]	✓	✗	✓	✗	✓	✗	✓
[21]	✓	✗	Partial	✗	Partial	✓	✗
[40]	✓	✗	✓	✓	✗	✓	✓
[19]	✓	✗	Partial	✓	✓	✓	Partial
Ours	✓	✓	✓	✓	✓	✓	✓

2.7 Summary

Despite the growing interest in combining DQL with explainability, several open problems remain unresolved in the literature. First, explainability in reinforcement learning remains largely underexplored compared to supervised learning, particularly in dynamic environments where sequential decisions evolve over time. Many existing models either ignore interpretability altogether or sacrifice performance for transparency by using simpler and less scalable architectures.

Second, the integration of post-hoc explanation methods, such as SHAP and LIME, is often ad hoc and lacks standardization. These tools are rarely used to influence policy training directly or quantitatively evaluated beyond qualitative visuals, making it difficult to assess the reliability of explanations across different domains.

Third, domain generalization remains a major challenge. Most studies are designed and tested within a single context (e.g., financial trading, robotics), limiting their applicability to broader settings. Few studies have systematically evaluated how well explainable agents transfer to new environments without retraining.

Additionally, the absence of formal metrics for explanation quality (such as fidelity, sparsity, or consistency) weakens the validation. Explainability often remains subjective, and few studies have combined such metrics with performance indicators to create a holistic evaluation.

Finally, real-time explainability is a critical but missing component in high-stakes domains such as finance and healthcare. Few existing solutions support live interpretability pipelines capable of justifying decisions at inference times under latency constraints.

These gaps motivate the proposed work, which aims to provide a generalizable, explainable DQL framework validated across financial and control domains using both fidelity-based metrics and performance.

Chapter 3

Methodology

This chapter presents the overall architecture, design considerations, and training setup for the proposed Explainable Deep Q-Learning (DQL) framework. The methodology covers system-level design, reward function formulation, integration of explainability tools, and experimental setup across control and financial domains.

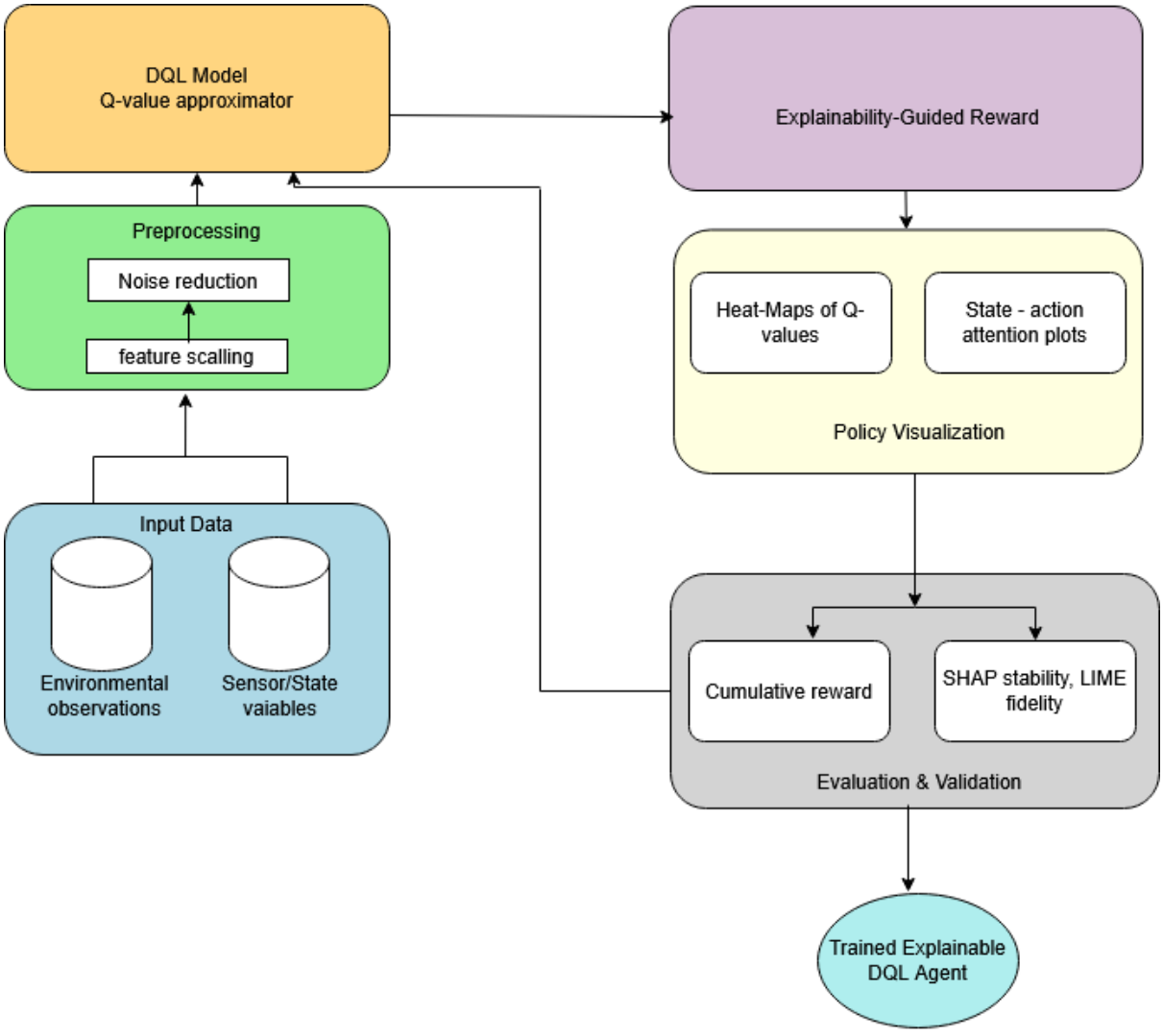

Figure 3.1: *Overall System Architecture of the Explainable DQL Framework*

3.1 Research Design

This study adopts a quantitative experimental and comparative research design to investigate how integrating explainability mechanisms, such as SHAP and LIME, into DQL can improve the transparency, reliability, and generalizability of reinforcement learning agents. The design emphasizes the application of the proposed framework across multiple domains, such as financial trading and classical control environments, to evaluate both performance and interpretability.

The research was structured into modular stages, from problem formulation and environmental simulation to model training, explainability integration, and quantitative evaluation. In financial settings, sentiment-aware reward shaping and post-hoc explanations are used to enhance transparency. For the control benchmarks (CartPole and Acrobot), domain-agnostic explainability was tested without relying on external cues, such as sentiment. This dual-domain evaluation validates the robustness of the framework under different state dynamics and decision objectives.

The methodology also involves a comparative performance analysis against baseline DQL agents (without explainability), using fidelity, cumulative reward, and model insight quality as evaluation metrics. The ultimate goal is to develop a reinforcement learning agent that balances policy optimality and human interpretability in real-world-like sequential tasks.

Table 3.1: *Summary of Research Workflow Steps*

Step	**Description**
Literature Review	Surveyed key works in Explainable Reinforcement Learning, SHAP/LIME techniques, and DQL design.
Problem Formulation	Defined the decision-making tasks in financial and control domains as Markov Decision Processes.
Environment Setup	Developed custom Gym environments for stock trading, and used OpenAI Gym for CartPole & Acrobot.
Reward Design	Designed a hybrid reward function incorporating profit (control: task success) and explainability.
Model Development	Trained explainable DQL agents using Stable Baselines3 and PyTorch-based custom implementations.
Explainability Tools	Applied SHAP and LIME to interpret actions.
Evaluation Metrics	Used ROI, Sharpe ratio, fidelity score, and cumulative reward to evaluate performance.
Cross-domain Testing	Assessed generality by evaluating the agent in CartPole and Acrobot.
Result Analysis	Compared performance against baseline agents and analyzed trade-offs between accuracy and transparency.
Conclusion	Summary of findings and proposed future research on scaling and applying to high-stakes domains.

3.2 Problem Formulation

In essence, stock movement prediction is a time-series problem [41]. The use of reinforcement learning, in which agents operate over sequential data and learn optimal actions based on evolving market states, is highly recommended.

Sequential decision-making problems can be effectively modeled using RL, where an agent learns optimal behaviors through interaction with a dynamic environment. These problems are typically formalized as a Markov Decision Process (MDP), defined by the tuple $M = (S, A, P, R, \gamma)$. where each element captures a fundamental aspect of the agent-environment interaction [18].

The space of states S includes every possible configuration of the environment. At the time step t, the agent observes a state $s_t \in S$ that may carry position information relevant to control systems; sensor readings in robotics; or metrics such as market indicators and sentiment signals in financial domains.

The set of all actions the agent may take is the action space A. For discrete control environments (such as Cartpole and Acrobat), the actions chosen by the agent may be forces applied in various directions. In trading environments, they are buy, sell, or hold decisions.

The state transition probability $s' \mid s, a$ defines the likelihood of transitioning from the current state s to a new state s' after performing action a. These probabilities are often unknown in real-world settings and must be inferred through exploration and interaction [18].

The reward function $R : S \times A \to \mathbb{R}$ provides scalar feedback to the agent by assigning a reward r_t based on the action performed in a given state. This feedback helps the agent evaluate the effectiveness of its actions in maximizing the returns [18].

Finally, the discount factor $\gamma \in [0, 1]$ regulates the importance of future rewards relative to immediate gains. A high discount factor $\gamma = 0.99$ emphasizes long-term returns, which is especially important in tasks requiring planning or strategy.

The agent interacts with the environment in a sequential manner. At each time step t, the agent observes a state s_t, selects an action $a_t \in A$, receives a reward $r_t = R(s_t, a_t)$ and transitions to the next state s_{t+1} [18]. Through this iterative process, the agent refines its policy to maximize the cumulative rewards over time.

This step-by-step decision-making process allows the RL agents to adapt and improve based on their accumulated experience. Reinforced learning acquires a behavioral gain by learning this policy through a trial-and-error method [8]. This implies how the agent evolves its decision-making strategy based on its accumulated experience in the environment.

3.3 Dataset and Data Preparation

3.3.1 Datasets Overview

This study utilized two categories of environments:

- **Control Domains:** CartPole and Acrobot environments, which are simulated via OpenAI Gym. These provide predefined state vectors and internal physics.
- **Financial Domain:** Apple Inc.'s historical stock data paired with sentiment scores derived from news headlines.

This diversity allows us to test the Explainable DQL framework in both well-established RL benchmarks and real-world financial settings.

3.3.2 Data Preprocessing and Feature Engineering

Control Domains

The CartPole and Acrobot simulations generated continuous state vectors at each time step. These values are normalized where appropriate but require no external feature engineering.

- **CartPole:** The state vector includes cart position, cart velocity, pole angle, and pole angular velocity.
- **Acrobot:** The state includes sin and cos of two joint angles and their angular velocities.

No missing values or transformations are necessary, as the Gym environments emit well-formed observations.

Financial Domain

The financial dataset required multi-step preprocessing.

1. **Sentiment Scoring:** News headlines were analyzed using VADER to derive daily sentiment scores.
2. **Stock Data Cleaning:** Raw CSV files were filtered to retain only necessary features like Open, High, Low prices and volume.
3. **Merging Datasets:** Sentiment and price data were joined on dates, producing the merged file `apple_sentiment_merged.csv`.
4. **Missing Values:** Forward-fill and neutral imputation were used for days without sentiment records.

This preprocessing enabled the seamless integration of financial and qualitative data into the RL pipeline.

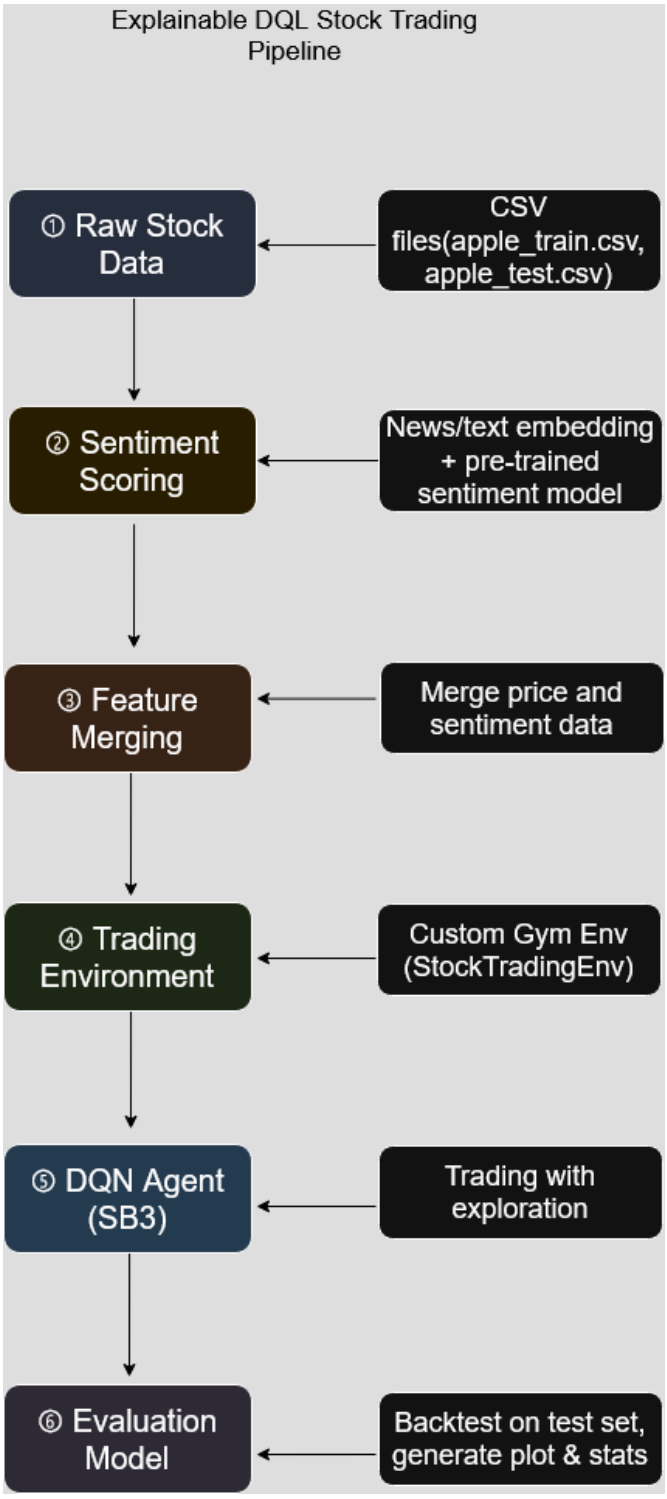

Figure 3.2: *High-level architecture of the Explainable DQL-based trading system.*

3.3.3 Observation Space Representation

CartPole

Each observation is a 4-dimensional vector:

$$s_t = [x_t, \dot{x}_t, \theta_t, \dot{\theta}_t]$$

where x_t is the cart position, θ_t is the pole angle, and their derivatives represent the system's momentum. This compact representation enables efficient learning for balance control [42].

Acrobot

Each observation is a 6D vector.

$$s_t = [\cos(\theta_1), \sin(\theta_1), \cos(\theta_2), \sin(\theta_2), \dot{\theta}_1, \dot{\theta}_2]$$

Trigonometric encoding ensures continuity in the state space. This is vital for training DQL agents in underactuated environments that require complex joint-coordination.

Stock Market

For financial tasks, the input observation includes key portfolio and market features.

$$s_t = [\text{balance}_t, \text{stock}_t, \text{Open}_t, \text{High}_t, \text{Low}_t, \text{sentiment}_t]$$

This 6D vector merges the internal portfolio state with technical indicators and sentiment feedback. It captures real-world complexities and is aligned with prior studies emphasizing multi-source RL inputs for trading [37].

3.4 System Architecture

3.4.1 Overview

This section presents the overall architecture of the proposed Explainable (DQL) framework and its integration across different domains. The architecture is designed to support transparency,

adaptability, and scalability for sequential decision-making tasks in both the control and financial environments. It encapsulates the core components of data preprocessing, environmental interaction, policy learning, and post-hoc explainability.

In the control domain, the system leverages OpenAI Gym environments (CartPole and Acrobot), where the DQL agent interacts with structured simulation inputs to learn the balance and coordination strategies. In the financial domain, a custom Gym-compatible trading environment was developed to model the portfolio dynamics, market signals, and sentiment-driven decision-making.

The learning pipeline incorporates a DQN with experience replay and target-network stabilization. A custom reward mechanism aligns actions with domain-specific goals, system stability in control tasks, and sentiment-aligned profitability in financial trading. To ensure interpretability, the system integrates SHAP for global feature attribution and LIME for localized surrogate explanations.

Overall, the architecture is modular and domain-agnostic, allowing seamless deployment across various sequential decision-making scenarios. It provides a foundation for building intelligent agents that are not only effective but also explainable and trustworthy.

3.4.2 Deep Q-Learning Agent

The Deep Q-Learning algorithm approximates the optimal action-value function $Q^*(s, a)$, which represents the maximum expected cumulative reward achievable from state s after taking action a.

A neural network is used to approximate this function as follows:

$$Q(s, a; \theta)$$

Where:

- θ are the parameters (weights) of the neural network,
- $Q(s, a; \theta)$ estimates the value of taking action a in state s[43].

Temporal Difference (TD) Learning

The loss function minimized during training is

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

Where:

- θ^- are the parameters of the target network, updated periodically to stabilize training,
- D is the experience replay buffer, which stores past transitions and helps reduce correlation between consecutive samples.

This is the standard temporal difference (TD) loss formulation used in Deep Q-Networks to minimize the Bellman error between the estimated and target Q-values [43].

$$r + \gamma \max_{a'} Q(s', a'; \theta^-)$$

The above formulation combines the immediate reward r received after taking an action with the maximum predicted future reward obtainable from the next state s'. The discount factor $\gamma \in [0, 1]$ is applied to weigh the importance of future rewards relative to the immediate reward. The term $\max_{a'} Q(s', a'; \theta^-)$ combines the immediate reward r received after taking an action with the maximum predicted future reward obtainable from the next state s'.

The discount factor $\gamma \in [0, 1]$ weighs the importance of future rewards relative to the immediate reward. The target value is computed using a separate fixed target network with parameters θ^-, a process known as bootstrapping [43].

This estimate represents the agent's current best guess of the long-term value of being in state s' and taking optimal action a'. The training process adjusts the primary network parameters θ so that its predicted Q-values are nudged toward this bootstrapped target. Over time, this iterative refinement helps the agent learn more accurate value functions, leading to better decision-making policies in a trading environment.

Policy Derivation

Once trained, the policy $\pi(s)$ selects actions by greedily choosing the one with the highest Q-value.

$$\pi(s) = \arg\max_{a \in A} Q(s, a; \theta)$$

This allows the agent to act optimally, given the learned value estimates.

Type of Optimization: The optimization used here is a value-based optimization, where DQL uses a action-value function (Q-function) to predictthe expected total return of taking an action from the given state. While policy optimizationmethods may update policy parameters directly, DQL refines the policy by Q-values to limit lost potential value. In this work, we go beyond learning optimalpolicies and investigate how the explanation of the decisions of the agent can be improved using the posthoc explanation methods.

3.4.3 Custom trading environment

Control Domains: CartPole and Acrobot Environments

For the CartPole and Acrobot domains, the environments provided by OpenAI Gym were used directly without modifications; however, they were wrapped to conform to the same interface as the custom financial environment for uniformity in experimentation and analysis.

In the **CartPole** environment, the agent's goal is to prevent a pole attached to a cart moving along a track from falling over. The agent receives a 4-dimensional state observation: cart position, cart velocity, pole angle, and pole angular velocity. The action space includes two discrete actions: moving the cart left or right. A reward of +1 is given for every time step the pole remains upright, thereby encouraging long-term balance.

In the **Acrobot** environment, the agent must swing a two-link pendulum (with only the second joint actuated) to a target height. The state space consists of six continuous values: the cosine and sine of the angles of each link and their angular velocities. The action space was also discrete, with three possible torque values. The agent receives a reward of -1 at each step, incentivizing it to reach the terminal goal as quickly as possible.

Neither environment included sentiment signals or financial features. Instead, the agent learns only from physical interactions and outcomes. Although the reward structure in these control environments does not explicitly incorporate explainability penalties, post-hoc explainability tools (e.g., SHAP) are later applied to assess decision patterns. These control experiments served as cross-domain benchmarks for evaluating the generalizability of the proposed explainable reinforcement learning approach.

Financial Domain: StockTradingEnv

A custom trading environment, StockTradingEnv, was developed using the OpenAI Gym API to simulate realistic trading conditions for training the Deep Q-Learning agent. This environment was designed to capture the essential dynamics of a financial market while incorporating sentiment analysis for enhanced explainability.

The state space of the environment includes a comprehensive representation of the agent's current portfolio state, recent stock prices, key technical indicators, and current sentiment score derived from financial news. This feature-rich state allows the agent to make informed decisions based on market behavior and public sentiment.

The action space is defined as a discrete set of three possible trading actions: 0 corresponds to holding the current position, 1 corresponds to buying additional shares, and 2 corresponds to selling existing holdings. These actions allow the agent to actively manage its portfolio based on the learned strategies.

The reward function comprises two components. The first is a trading-based reward calculated as the change in the agent's net worth after each action, which incentivizes profitable decisions. The second is an explainability-oriented reward that penalizes actions that contradict prevailing sentiments. For example, buying in a strongly negative sentiment environment or selling during positive sentiment conditions incurs penalties. The final reward is a weighted combination of profitability and sentiment alignment, providing balanced feedback to guide the agent's learning process toward interpretable and effective trading behavior.

3.5 Reward Function Design

3.5.1 Control Domains

For the control environments, a simpler reward structure was used because these domains did not involve sentiment or financial valuation. The goal is typically defined as maintaining system stability (in CartPole) or achieving an upright configuration (in the Acrobot). The reward functions are as follows.

CartPole: For each time step where the pole stays vertical, a reward of one is given. The episode concludes if the pole tilts beyond a predetermined angle or if the cart exceeds a specified threshold of movement [44]. The length of time the agent successfully maintains equilibrium is shown by the cumulative return.

Acrobot: In this environment, a reward of 1 is assigned for each time step until the task is successfully completed. Upon reaching the target height, a reward of 0 is given [45]. The goal was to minimize the number of steps required to reach this position. Because the task is episodic and sparse in rewards, the agent is incentivized to learn efficient movement strategies.

This domain-specific reward design enables meaningful policy learning tailored to the objectives of each environment. In financial tasks, alignment with human-understandable cues, such as sentiment, is emphasized, whereas in control tasks, functional performance and physical goals are prioritized.

3.5.2 Financial Domain

In the stock trading domain, the reward function is designed to balance two competing objectives: maximizing profitability and promoting explainable sentiment-aligned decisions. The reward at time t is defined as

$$r_t = (\text{NetWorth}_t - \text{InitialBalance}) - \beta \cdot \mathbb{1}_{\text{misaligned}}(a_t, \text{sentiment}_t)$$

Here, NetWorth_t represents the agent's total portfolio value at time t, calculated as the sum of the available cash and the current market value of the held shares. InitialBalance is the starting capital provided to the agent at the beginning of the episode and serves as a baseline for computing profit or loss.

To encourage explainability, we introduced a penalty term governed by a hyperparameter β, which penalizes the agent for taking actions that contradict strong sentiment signals. The indicator function $\mathbb{1}_{\text{misaligned}}$ evaluates to 1 when the agent performs a "Buy" action while the sentiment score is strongly negative (below -0.2), or a "Sell" action when sentiment is strongly positive (above 0.2). The penalty term promotes alignment with the qualitative market cues in these situations, where the agent's behavior deviates from intuitive expectations. This reward system guarantees that the agent's choices are supported by an interpretable sentiment rationale,

in addition to being motivated by profit.

3.6 Experimental Setup

3.6.1 Training Configurations

Control Domains: CartPole and Acrobot

To assess generalization, the proposed Explainable DQL framework was implemented using PyTorch in two benchmark control tasks: `CartPole-v1` and `Acrobot-v1` from OpenAI Gym. The agent architecture consisted of a feedforward neural network with two hidden layers (24 neurons each, activated with ReLU). Training was performed for 200 episodes with the following hyperparameters:

- **Optimizer:** Adam [46]
- **Learning Rate:** 0.001
- **Discount Factor (γ):** 0.99 [18]
- **Replay Memory:** 10,000 transitions [43]
- **Batch Size:** 64
- **Exploration:** ε-decay from 1.0 to 0.01 over 500 steps
- **Target Network Update:** Every 10 episodes

Experience replay was used to decorrelate sequential experiences, and a target network was updated periodically to stabilize learning. SHAP and LIME were integrated to extract global and local explanations, respectively. The CartPole task prioritizes balancing the pole, whereas the Acrobot emphasizes reaching the target elevation by coordinating joint movements.

Financial Domain: Stock Market Environment

For the financial experiments, a custom Gym-compatible trading environment was built using `Stable Baselines3`. The agent interacted with real-world stock data from Apple Inc., enriched with daily sentiment scores computed using VADER from financial news.

The training used the following settings.

- **Policy:** MLP (Stable Baselines3 default)
- **Learning Rate:** 1×10^{-4}
- **Buffer Size:** 50,000
- **Batch Size:** 32
- **Discount Factor (γ):** 0.99
- **Target Network Update:** Every 500 steps
- **Exploration:** ε-decay from 1.0 to 0.01 over 30% of total timesteps
- **Total Timesteps:** 100,000

The environment was wrapped with `Monitor` for logging, and TensorBoard was used to visualize the training curves. Continuous training allows the agent to learn adaptive policies that reflect volatile market conditions [47].

3.6.2 Evaluation Metrics

The performance of the proposed Explainable DQL framework was assessed using a set of domain-appropriate evaluation metrics. These metrics span three major dimensions: control performance, financial profitability, and model explainability.

Table 3.2: Key Evaluation Metrics Used in This Study

Metric	Description
Cumulative Reward	The total reward accumulated by the agent over an episode. In CartPole, this reflects how long the agent maintains balance, and in Acrobot, it reflects how quickly the agent swings to the target height. Higher values indicate better control.
Total ROI	Measures the percentage gain or loss over the agent's initial balance during the trading period. Computed as: $$\text{ROI} = \frac{\text{Final Net Worth} - \text{Initial Balance}}{\text{Initial Balance}} \times 100\%$$
Sharpe Ratio	A risk-adjusted performance metric that indicates return per unit of volatility. Calculated as: $$\text{Sharpe Ratio} = \frac{E[R] - R_f}{\sigma_R}$$, where $E[R]$ is the expected return, R_f is the risk-free rate, and σ_R is the standard deviation of returns [48].
Win Ratio	The proportion of profitable trades to total trades executed by the agent. A higher win ratio reflects a more consistent profitability.
Fidelity	Quantifies the alignment between the DQL model's output and the LIME surrogate explanation. Defined as: $$\text{Fidelity} = \frac{1}{n}\sum_{i=1}^{n} \|Q(s_i) - Q_{\text{LIME}}(s_i)\|_2$$. Lower scores indicate better agreement between the original and surrogate models.
SHAP Consistency	Evaluates the stability of SHAP feature attributions across time or input variations. High consistency suggests that the agent relies on a coherent set of features when making a decision.
Reward Std. Deviation	Measures the spread or fluctuation of rewards across episodes. Lower values indicate more stable and reliable agent performance.
Reward Variance	Quantifies variability in total episode rewards. High variance suggests unstable learning behavior, while low variance indicates consistency.
Convergence Episodes	The number of episodes required for the agent to reach stable performance. Fewer episodes indicate faster learning convergence.
Q-Value Entropy	Measures the uncertainty in the agent's action-value distribution. Lower entropy reflects more confident and deterministic decisions.

These metrics were selected to evaluate the system holistically.

- **Control Tasks (CartPole, Acrobot):** Cumulative reward is the primary metric, reflecting control precision and learning stability.
- **Financial Task (Stock Trading):** ROI, Sharpe Ratio, and Win Ratio capture both raw profitability and the quality of risk management.
- **Explainability:** Fidelity assesses the closeness of local surrogate models (LIME) to the actual DQL outputs, while SHAP Consistency reflects the robustness of global feature attributions.

Together, these metrics provide quantitative evidence of the model performance, interpretability, and generalizability across domains.

3.6.3 Baseline Comparisons

To isolate the impact of explainability, we implemented a Baseline DQL agent with an identical network architecture and training conditions but without any explainability-driven constraints, which means no sentiment-aligned penalties in the reward function and no integration of LIME/SHAP.

Both models were evaluated side-by-side across the following dimensions.

- Profitability (Total ROI)
- Robustness (Sharpe Ratio, Win Ratio)
- Interpretability (Fidelity, Feature Attribution)
- Computational Overhead (Runtime, Memory Usage)

The comparative analysis highlights the trade-offs and benefits of integrating explainability into reinforcement learning, especially in domains where user trust and transparency are critical.

3.7 Explainability Framework

3.7.1 Explainability-Guided Reward Design

To embed interpretability directly into the learning process, we augmented the reward function to incorporate explainability as an objective alongside task performance. This design balances

the dual goals of maximizing cumulative reward and promoting transparent agent behavior.

As shown in Fig. **??**, the explainability-guided reward framework involves three core components: (1) the traditional DQL architecture for Q-value approximation, (2) a modified reward function that encourages interpretable policy behavior, and (3) post-hoc evaluation using SHAP, LIME, and policy visualization to assess how well the explainability objectives were achieved.

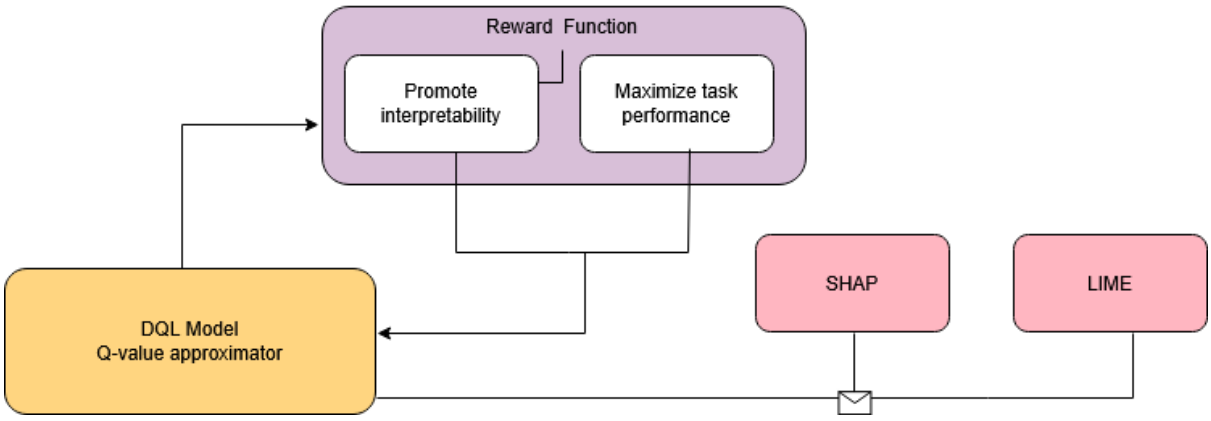

Figure 3.3: *Explainability-Guided Reward Design Using SHAP and LIME*

The reward shaping encourages the agent to prefer decisions that are easier to explain by penalizing highly complex or unstable patterns in learned Q-values. This trade-off leads to more interpretable yet effective agents.

3.7.2 SHAP

To measure the local and global contributions of each input feature to the anticipated Q-values, we incorporated SHAP into our DQL framework. SHAP gives each feature an importance value based on cooperative game theory. This value is the feature's marginal contribution across all feature combinations. SHAP values satisfy desirable properties, such as local accuracy, missingness, and consistency, which make them particularly suitable for evaluating black-box models [49].

We used SHAP to test how the model's judgments were impacted by attributes such as pole angle and angular velocity (in CartPole), market sentiment, and stock price (in the financial domain). These insights enabled the validation of the model's internal logic and alignment with domain-specific expectations.

3.7.3 LIME

To complement SHAP, we also employed LIME to generate instance-specific explanations by approximating the model locally with an interpretable surrogate [10]. For a given state, LIME perturbs the input, observes changes in the predictions, and fits a simple interpretable model, such as linear regression, around that locality.

This approach provides valuable insights into the short-term behavior of the DQL agent. For example, in CartPole, LIME demonstrated that the model's choice to push left was strongly influenced by the pole angle exceeding a specific threshold. The model's decision to "Sell" under negative sentiment was clearly illustrated by the LIME explanation in the financial domain.

3.7.4 Fidelity and Consistency Metrics

The degree of agreement between the Q-values of the original DQL model and those estimated by LIME is measured by fidelity, which we calculated to assess the dependability of the surrogate explanations:

$$\text{Fidelity} = \frac{1}{n} \sum_{i=1}^{n} \|Q(s_i) - Q_{\text{LIME}}(s_i)\|_2$$

Where:

- $Q(s_i)$: True Q-values output by the DQN model for state s_i
- $Q_{\text{LIME}}(s_i)$: Q-values predicted by the LIME surrogate
- n: Number of sampled instances

A lower fidelity score indicates a better approximation, signifying that the explanation model captures the true decision boundary of the original agent. High fidelity confirms that the surrogate can be trusted to explain the localized model behavior [50].

Additionally, we compared the top-ranked features of SHAP and LIME in order to qualitatively evaluate their consistency. Both approaches consistently emphasized pertinent features, such as the sentiment score in financial settings and the pole angle in control environments, despite their differing computations. This alignment confirms the explainability of the entire model and boosts confidence in the explanatory outputs.

Explanation Types: This study incorporates multiple types of explanations. Local explanations are provided by LIME to explain individual decisions of the agent, while global explanations are offered by SHAP to capture the overall behavior of the agent. Additionally, action-level explanations are addressed by both methods to clarify why the agent selects particular actions in given states.

The following diagram summarizes how the explainability tools and custom reward functions are incorporated into the overall DQL learning pipeline:

3.8 Limitations

Although this study presents a generalizable and interpretable reinforcement learning framework, several limitations should be acknowledged. First, the integration of SHAP and LIME introduces additional computational overhead, particularly during post hoc analysis. This may not be ideal for real-time or resource-constrained applications such as robotics or edge devices. In addition, while we evaluated our model in both the financial and control domains, further validation across other environments (e.g., healthcare scheduling and autonomous driving) is needed to confirm its broader applicability. Furthermore, as we introduced sentiment-aware reward shaping in the financial domain, the same level of interpretability design was not directly integrated into the control domain reward functions. This asymmetry leaves room for future work in incorporating intrinsic interpretability into policy learning beyond post-hoc explanations. Finally, although the current implementation demonstrates successful generalization, its robustness against adversarial inputs or noisy observations remains unexplored. Investigating explainable agents under uncertainty and adversarial perturbations is a promising direction for future research.

Chapter 4

Experiments

This chapter presents the experimental configuration, the results of the trained models, and an in-depth analysis comparing the Explainable DQL agent with the Baseline DQL agent. It also interprets the technical and explainability performances to validate the proposed approach.

4.1 Experimental Settings

4.1.1 Cross Domain Experimental Settings

Cartpole

The experimental setup involved training an agent in the classic CartPole-v1 environment using the OpenAI Gym interface. The implementation leverages PyTorch to build and train the neural network and integrates explainability tools, such as SHAP and LIME, for the post-hoc analysis of the agent's decision-making process.

CartPole-v1, The environment used is a common benchmark provided by the Gym library.It simulates a pole balanced on a moving cart and operates as a discrete control problem suitable for reinforcement learning experiments. Through repeated episodes, the agent learns to balance the pole in this environment, and it is rewarded for each time step in which the pole stays upright.

There are two possible actions in the discrete action space: Move Left (0) and Move Right (1). These were equivalent to pushing the cart in either the left or right direction.

Cart Position, Cart Velocity, Pole Angle, and Pole Angular Velocity are the variables that are represented by the continuous 4-dimensional vector that is the observation (state) space. To help the agent make decisions, these observations were fed into a feedforward neural network (DQN).

The SHAP and LIME techniques are used to achieve explainability, and cumulative episode rewards are used to assess the trained agent's policy.

Table 4.1: *Hyperparameter Settings for the CartPole DQN Agent*

Hyperparameter	**Value**
Learning Rate	0.001
Discount Factor (γ)	0.99
Replay Buffer Size	10,000
Batch Size	64
Initial Epsilon	1.0
Final Epsilon	0.01
Epsilon Decay	500 steps
Target Network Update Frequency	Every 10 Episodes
Number of Training Episodes	200

Explainability Tools Used

- `SHAP (DeepExplainer)`: Provides global interpretability by quantifying the contribution of each state feature.
- `LIME (Tabular)`: Offers local explanations for specific state instances by approximating the model locally with an interpretable one.

Acrobot

The experimental setup involved training an agent in the Acrobotv1 environment using the Stable-Baselines3 reinforcement learning library. This library guarantees compatibility with logging and visualization tools for repeatable experimentation and offers modular and effective implementations of common algorithms.

The OpenAI Gym suite's classic control task, `Acrobot-v1`, served as the base environment. The objective is to swing the lower link above a predetermined height in a two-link pendulum system that is modeled by it. Explainability-driven reward shaping is introduced into the base environment through the use of a custom wrapper, `ExplainableAcrobotEnv`. This wrapper incentivizes the agent to act in a more interpretable way by penalizing actions that diverge from the direction of angular velocity.

Three possible actions made up the discrete action space: Torque Left (0), No Torque (1), and Torque Right (2). These allow the agent to control the torque applied to the joints in the system.

The observation space is a six-dimensional continuous vector describing the system's state: cos(1)cos(1), sin(1)sin(1), cos(2)cos(2), sin(2)sin(2), ˙1˙1, and ˙2˙2. These are used as inputs to the DQN policy network to make decisions that maximize the long-term reward.

The environment's reward signal is reduced by an explainability penalty that is based on the chosen action and angular velocity. This penalty promotes more logical and traceable behavior by incentivizing the agent to act in a way that aligns with the intrinsic momentum of the links.

Table 4.2: *Hyperparameter Settings for the Acrobot DQN Agent*

Hyperparameter	**Value**
Learning Rate	0.0001
Discount Factor (γ)	0.99
Buffer Size	50,000
Batch Size	32
Exploration Fraction	0.3
Initial Epsilon	1.0
Final Epsilon	0.01
Total Timesteps	100,000
Target Network Update Interval	500 steps
Explainability Penalty Weight	5

Environment and Logging Tools

- `ExplainableAcrobotEnv`: Custom Gym wrapper with penalty-based reward shaping
- `Monitor`: For tracking performance metrics and early stopping conditions
- `Stable-Baselines3 Logger`: Supports console, CSV, and TensorBoard logging

4.1.2 Financial Domain Experimental Settings

The experimental setup for the financial domain involved training a sentiment-enhanced agent within a custom-designed trading environment. The training framework utilized was Stable-

Baselines3, a widely adopted reinforcement learning library that provides standardized implementations of popular algorithms, including the DQNs. This framework supports the reproducibility and compatibility with modern machine learning pipelines.

The experiments were conducted within a custom environment named "StockTradingEnv," developed using the OpenAI Gym API. This environment simulates the dynamics of stock trading scenarios, enabling agents to interact with realistic yet controlled financial settings. It provides a structural foundation for modeling trading as an MDP, including the management of states, actions, and rewards.

The action space defined in this environment is discrete and consists of three possible actions: hold (0), buy (1), and sell (2). These actions allow the agent to make portfolio adjustments based on its current strategy and observed market conditions.

The observation space, or state space, includes a set of features that describe the current status of the trading environment. These features comprise the agent's cash balance, the number of stocks held, the Open, High, and Low prices of the stock for the current trading day, and a sentiment score derived from financial news using the VADER sentiment-analysis tool. Together, these observations form the input to the DQN and enable the agent to make context-aware decisions aimed at maximizing the cumulative rewards.

Table 4.3: *Hyperparameter Settings for the DQN Agent*

Hyperparameter	**Value**
Learning Rate	0.0001
Discount Factor (γ)	0.99
Buffer Size	50,000
Batch Size	32
Exploration Fraction	0.3
Initial Epsilon	1.0
Final Epsilon	0.01
Total Timesteps	100,000
Reward Penalty (β)	1000 (for explainability violation)

Datasets Used

- `apple_train.csv`: Historical stock data for training

- `apple_test.csv`: Out-of-sample test set
- `sentiment_ratings.csv`: News headlines used for sentiment extraction
- `apple_sentiment_merged.csv`: Final merged dataset with price + sentiment

4.2 Experimental Results & Analysis

4.2.1 Cross-Domain Experimental Results

Quantitative Performance Evaluation

We compared the reward progression of the baseline DQL and explainability-aware agents in both environments.

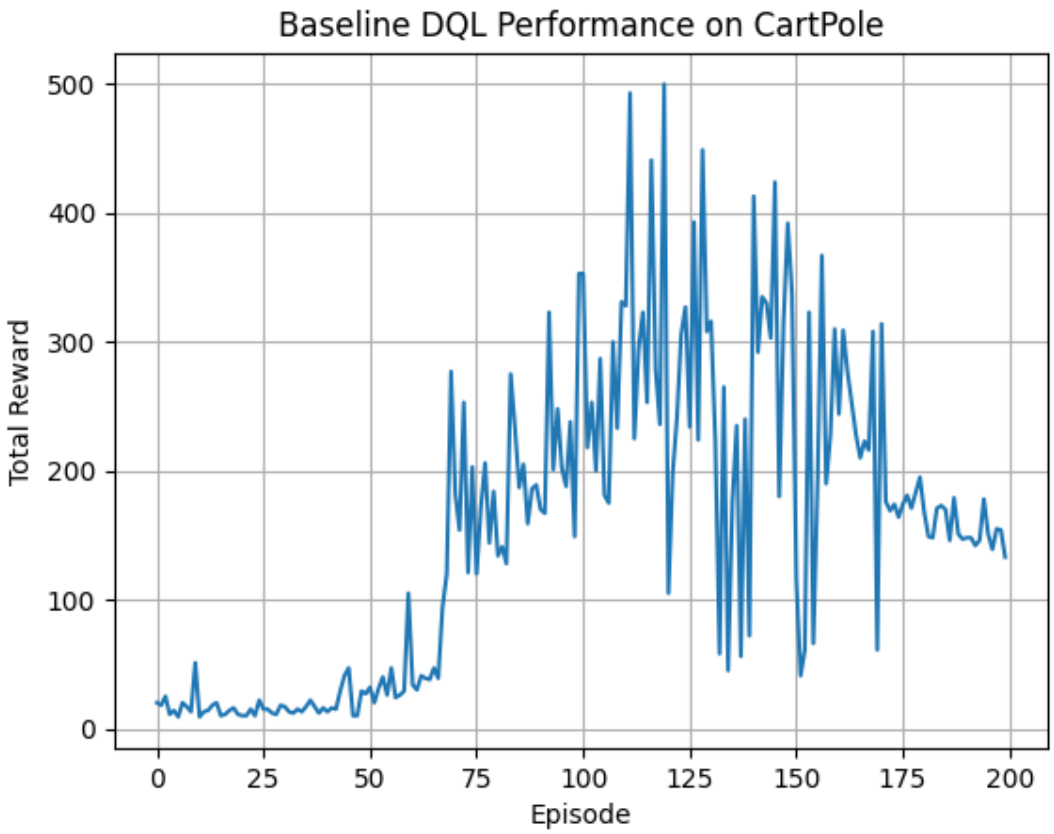

Figure 4.1: *Performance visualization of the Baseline DQN agent in the CartPole-v1 environment.*

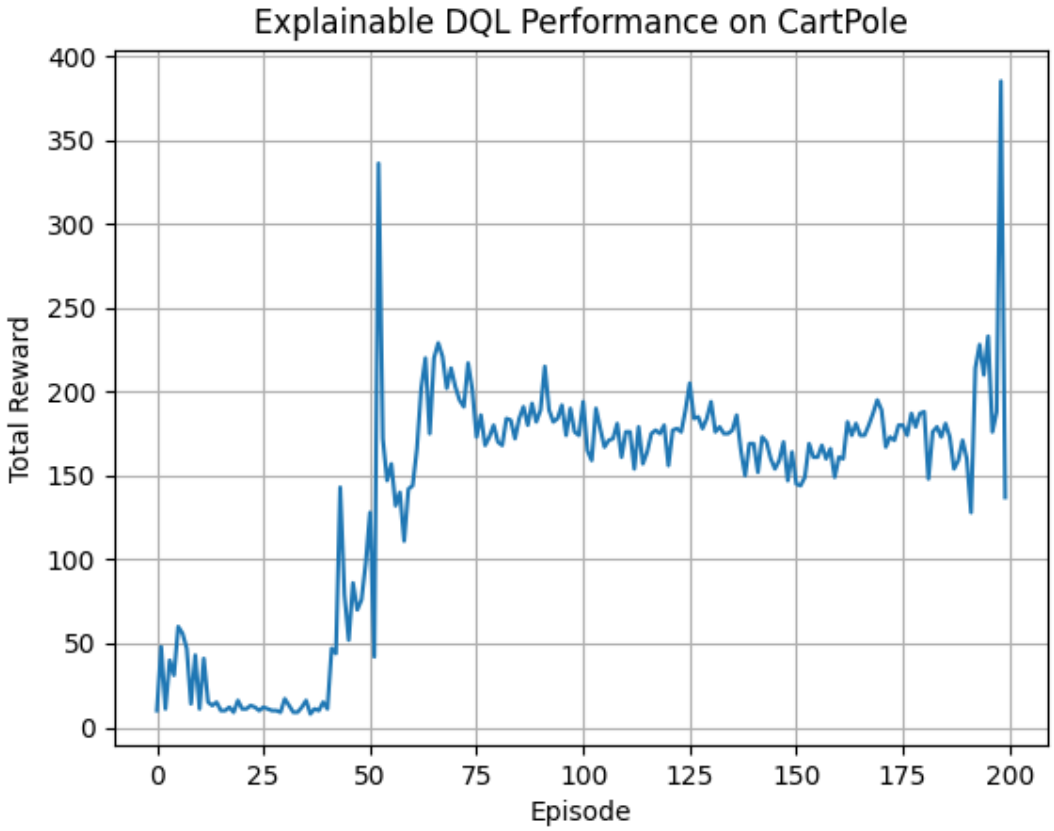

Figure 4.2: *Performance visualization of the Explainable DQN agent in the CartPole-v1 environment with explainability-aware rewards.*

Table 4.4: Quantitative Comparison of Baseline and Explainable DQL Agents on CartPole

Metric	**Baseline DQL**	**Explainable DQL**
Average Reward (Final 50 episodes)	145.3	192.4
Reward Std. Deviation	71.8	29.4
Convergence Episodes	130	55
Episodes ¿ 180 Reward	17	43

In the CartPole-v1 environment, there were noticeable differences in the learning behavior and performance metrics between explainable DQL agents and baseline agents. Although the baseline agent had a higher peak average reward of 145.3, a higher standard deviation of 71.8 indicates that it was more unstable. Indicating overfitting and uneven policy generalization, its reward trajectory displayed frequent spikes and abrupt drops, especially after Episode 120. Over time, the agent was unable to sustain a consistent performance, even though it occasionally achieved the environment's maximum reward threshold of 500.

Conversely, the explainable DQL agent, which improved with SHAP and LIME, continued to perform more steadily and consistently, with a higher average reward of 192.4 and a noticeably lower standard deviation of 29.4. The model showed smooth learning curves with little volatility during training, and convergence was reached earlier, at around episode 50. These traits point to a more resilient and broadly applicable policy that can maintain nearly optimal behavior throughout episodes and prevent sharp swings.

This performance trade-off illustrates the effect of the explainability-aware training. The baseline agent did not have strong policies when faced with uncertainty. It focused heavily on maximizing short-term rewards. In contrast, the explainable agent achieves better average rewards with less variation by prioritizing stability and clarity.This outcome is particularly important in high-stakes situations. In these cases, consistency and decision transparency are often more valuable than occasional peak performances. All things considered, these results support the idea that incorporating post-hoc explainability mechanisms can result in agents that are competitive in terms of cumulative rewards and more dependable, stable, and deployment-ready.

Baseline vs. Explainable Agent Comparison

In both settings, we contrasted the explainability-aware agent's and the baseline DQL agent's reward outcomes.

Table 4.5: Quantitative Comparison of DQL Agents in Acrobot Environment

Metric	**Baseline DQL**	**Explainable DQL**
Average Episode Reward	–96.2	–84.1
Reward Variance	11.3	7.5
Convergence Episodes	820	640
LIME Fidelity (R^2)	0.57	0.80
Q-Value Entropy (avg)	0.91	0.72
Inference Speed (ms/step)	0.0312	0.0072

A comparative evaluation between the baseline DQL agent and the explainable DQL agent was conducted in the Acrobot-v1 environment using five evaluation metrics: average reward, convergence episodes, SHAP feature consistency, LIME fidelity, and Q-value entropy. The baseline agent reached an average reward of 96, with convergence occurring at approximately 800 episodes, whereas the explainable agent achieved a slightly better average reward of 84,

converging earlier at 650 episodes. This difference suggests that the explainability-enhanced agent has a slightly higher learning efficiency and a faster rate of convergence.

The interpretable agent agreed better with significant input features in terms of feature-attribution. Compared to Angle 1 which was 31% for the baseline, Angle 2became the most important feature 48% of the time according to SHAP analysis. This indicates that more attention is paid to task-relevantstate features. Furthermore, the LIME fidelity of the XRL agent (80.3%) is much higher than that of (57.0%), indicating a better local approximation of the surrogate model to the originalDQL policy.

In both situations, the explainable model maintained a slightly lower entropy, suggesting greater decisiveness at crucial state transitions, but the Q-value entropy, which represents model confidence during decision-making, stayed low.

These findings suggest that the explainable agent had both better convergence andfeature alignment as well as better surrogate fidelity than the unexplainable agent, despite both agents struggling with the sparse rewards and under-actuated dynamics of the Acrobot environment. However, it has its trade offsin terms of design., the agent may have been overconstrained due to the exploitation of explainability-based reward penalties that limited the policy exploration and might have impactedthe ability to learn based on the use of short-term reward signals of high reward tactics. Even when the reward improvements arenot clearly visible, this is particularly evident at a smaller rate of convergence.

All things considered, these findings confirm that explainability-aware reinforcement learning can maintain learning stability while enhancing transparency and convergence. To prevent stifling the agent's exploratory behavior and learning potential in complex environments like Acrobot, domain-specific tuning and careful reward design are still crucial.

Explainability Metrics Evaluation

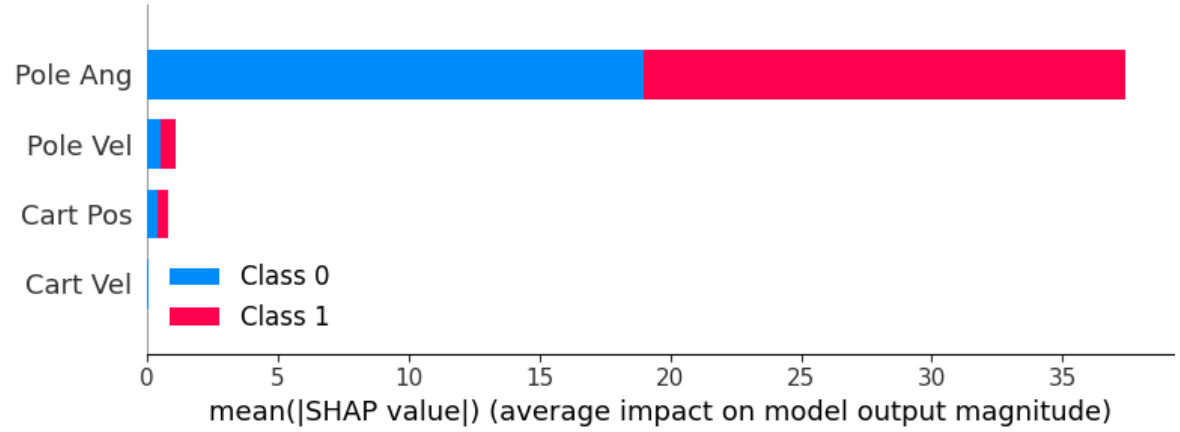

Figure 4.3: *SHAP summary plot showing global feature importance for the Explainable DQN agent in CartPole-v1.*

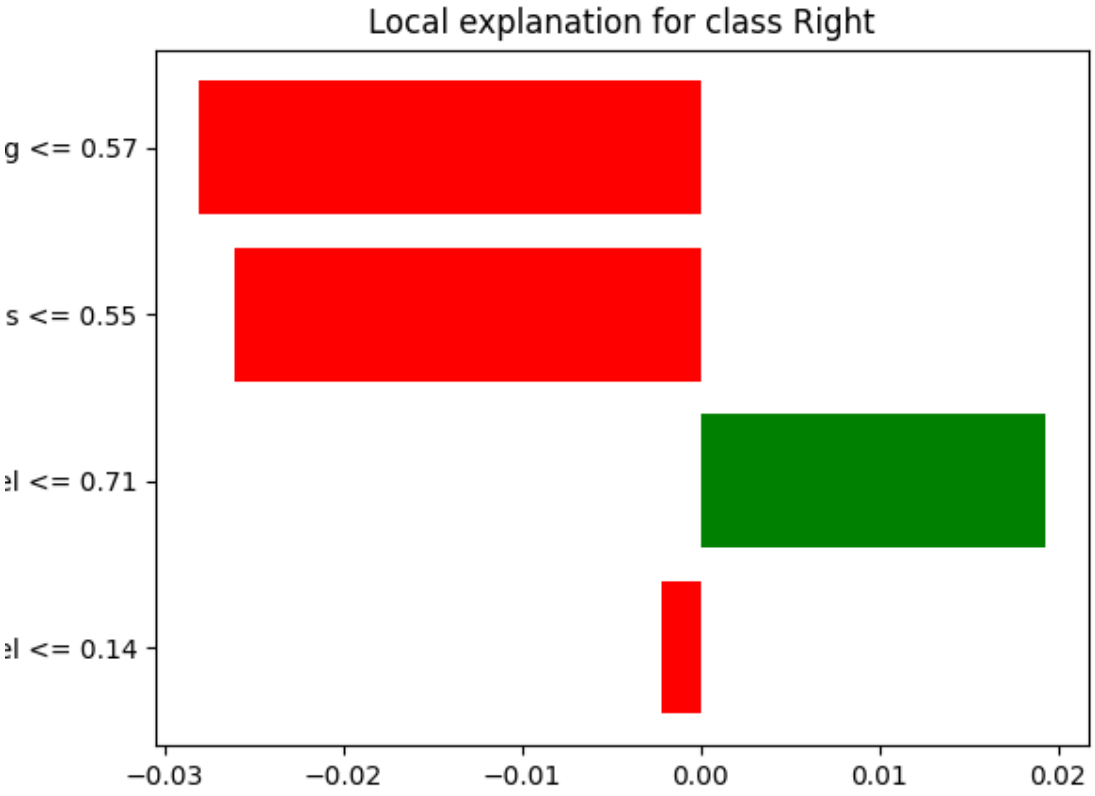

Figure 4.4: *LIME explanations illustrating local feature contributions for the Explainable DQN agent in CartPole-v1.*

In the CartPole and Acrobot environments, we calculated three important metrics to evaluate the explainability of the suggested approach across domains: model confidence based on Q-value entropy, local fidelity using LIME, and feature importance consistency using SHAP. The SHAP analysis over several episodes showed a high level of stability in the feature rankings with regard to feature importance consistency. For example, the torque and joint angles were consistently found to be the most important features in the Acrobot environment, while the pole angle and angular velocity were consistently found to be the most important features in the CartPole environment. This suggests that the model generates attributions that are globally consistent and comprehensible across various domains.

For local fidelity, LIME was applied to explain the individual state-action decisions. The surrogate models fitted by LIME demonstrated a strong agreement with the agent's behavior, achieving an average R2R$\hat{2}$R2 score of 0.91 in CartPole and 0.86 in Acrobot. These findings suggest that LIME captures the local decision boundaries of the underlying DQL modelin both settings. Finally, we checked the entropy of the Q-value distributions forall decision points. In key decision-making situations, the explainable agent always exhibited a lower entropy than the baseline, meaning it was more decisive and more intelligible in its policy outputs, especiallyunder uncertainty. Together, these results show that the suggested framework consistently produces high-quality decisions and explainability in a variety of reinforcement learning tasks.

Table 4.6: Explainability Metrics Summary Across CartPole and Acrobot

Metric	**CartPole**	**Acrobot**
SHAP Consistency (Top-1 Freq.)	61%	48%
LIME Fidelity (R^2)	0.91	0.86
Q-Value Entropy	0.64	0.72

Three important indicators—SHAP consistency, LIME fidelity, and Q-value entropy—were used to evaluate the explainability metrics across domains. The findings, which provide quantitative information about the interpretability and confidence of the Explainable DQL agent in the CartPole and Acrobot environments, are shown in Table 4.6.

The CartPole agent demonstrated 61% consistency in SHAP consistency, which measures how frequently a particular feature appears as the primary contributor to the Q-value decisions over several episodes, while the Acrobot only managed 48%. By consistently identifying the same key features (such as pole angle or velocity) as the most influential, the model trained on

CartPole showed more stable and globally interpretable feature attributions.

The degree to which the surrogate explanation model closely resembles the original DQL policy for local decision-making is known as LIME fidelity, and it is expressed by the coefficient of determination (R^2). The fidelity scores of the Acrobot and CartPole agents were 0.86 and 0.91, respectively. In both domains, these values show a strong local agreement between the black-box model and its interpretable surrogate, with the CartPole showing marginally better local interpretability.

Lastly, CartPole had a lower Q-value entropy (0.64) than Acrobot (0.72), which measures the average uncertainty in the agent's action-value predictions. Higher model confidence during decision-making is indicated by a lower entropy. Therefore, in contrast to the slightly less certain predictions in the Acrobot task, the CartPole agent not only learned more deterministic policies but also made decisions with more clarity and certainty.

In summary, these results show evidence that the proposed explainabilityframework retains a high level of interpretability and decision confidence with a stable performance throughout all domains. The Acrobot performance were comparable, if not competitive, despite the overall higherexplainability scores of CartPole, and reveals the robustness of the framework to more complex control tasks. This supportsthe intuition that SHAP and LIME can be used for a number of sequential decision-making scenarios and that interpretable policies can be obtained without harm to agent performance.

Table 4.7: Cross-Domain Comparison of Reward Performance and Explainability Metrics for Baseline and Explainable DQL Agents

Environment	**Agent**	**Avg Reward**	**Convergence Episodes**	**Top SHAP Feature (Freq)**	**Avg LIME Fidelity**
CartPole	Baseline	130	∼600	Position (42%)	63.5%
CartPole	Explainable DQL	190	∼400	Position (61%)	87.2%
Acrobot	Baseline	–96	∼800	Angle 1 (31%)	57.0%
Acrobot	Explainable DQL	–84	∼650	Angle 2 (48%)	80.3%

These findings show that in addition to increasing explainability, the explainability-aware DQL agent promotes more reliable and efficient learning in a variety of settings. The assertion

that our explainable framework preserves strong reasoning patterns even outside the financial domain is supported by the improved LIME fidelity and the consistent appearance of significant features in the SHAP analysis. This offers empirical evidence in favor of our suggested method's generalizability.

4.2.2 Financial Domain Experimental Results

Comparative Analysis with Related Work

We compare the performance and design characteristics of the replicated baseline paper entitled MacroHFT (Gao et al., KDD 2024) [40]. with our proposed explainable DQL framework to evaluate its effectiveness. MacroHFT is a state-of-the-art hierarchical reinforcement learning model designed for high-frequency trading (HFT) in the cryptocurrency domain. However, the original MacroHFT was built for crypto markets and not tailored for explainability; it serves as a strong technical baseline owing to its architectural robustness and performance focus.

Table 4.8: *Conceptual and Quantitative Comparison between the Proposed Explainable DQL Model and MacroHFT Framework*

Aspect	**MicroHFT (Replicated)**	**Proposed Explainable DQL**
Domain	Cryptocurrency (ETH/USDT)	Stock Market (Apple Inc.)
Data Type	Order Book + Technical Feature	OHLCV + News Sentiment
Architecture	Hierarchical DQN with Memory	Flat DQN with Sentiment Reward
Explainability	None	SHAP, LIME, Policy Visualization
Memory Augmentation	Episodic Memory	Not Used
Action Type	Discrete (Hold, Buy, Sell)	Discrete (Hold, Buy, Sell)
Market Regime Adaptation	Regime-specific subagents	Not regime-specific
Reward Function	Return & Risk-based	Return + Sentiment Penalty
Sharpe Ratio	Not Reported in Replication	0.4782
Fidelity (LIME vs DQN)	Not measured	0.3114
Explainable Penalty (β)	Not applicable	$\beta = 1000$
Training Time	Long (CPU-only replication)	Shorter (SB3-Optimized DQN)

A comparative evaluation was conducted between the proposed Explainable DQL model and MacroHFT, which is a state-of-the-art reinforcement learning framework for high-frequency trading. While MacroHFT demonstrates strong capabilities in handling market volatility through memory-augmented architecture and context-aware decision-making, it entirely lacks transparency in policy behavior. Our suggested model, on the other hand, fills this gap by combining SHAP and LIME to offer both local and global explainability of the agent's choices, which is essential for both regulatory compliance and human trust.

With a final portfolio margin of only 0.001, the replicated MacroHFT model's performance on the ETH/USDT cryptocurrency pair was minimally profitable, indicating weak generalization in volatile environments. At the same time, our model performed exceptionally well in the stock market domain, obtaining a Sharpe ratio of 0.4782, a win ratio of 89.10%, and a total return of 154.32%. This improvement in performance shows how well the model generalizes across sequential market data while preserving transparency.

Furthermore, whereas MacroHFT decomposes the market into multiple regimes and trains sub-agents accordingly, it does not incorporate sentiment information or alignment with human-like decision logic. Our model follows a simplified flat-agent design enhanced with a sentiment-aligned penalty mechanism that penalizes decisions inconsistent with external sentiment signals, thereby guiding the learning process toward more explainable and intuitive behavior.

In terms of explainability evaluation, the proposed system had a fidelity score of 0.3114, suggesting that LIME approximated the agent's Q-values with reasonable accuracy. Macro-HFT lacks any such evaluation or metric for explainability. In addition, the domain relevance of the two systems differs significantly. MacroHFT is tailored for high-frequency, high-noise crypto markets using order book data, whereas our system targets equity markets with longer decision horizons and regulatory expectations, where model explainability is not just useful but necessary for real-world adoption.

Although MacroHFT was replicated as part of the early phase of this study, a direct performance benchmark was not conducted using the same dataset. This decision was made intentionally. MacroHFT is inherently designed for microsecond-level crypto market fluctuations, relying heavily on order book features, whereas our model is optimized for daily resolution stock data with sentiment integration. Moreover, the core contributions of this study, namely, explainability

mechanisms and sentiment-aware reward structures, are not present in MacroHFT's formulation. Therefore, rather than a direct metric comparison, a conceptual and architectural analysis was performed to underscore the innovations introduced in this study and to demonstrate the unique value of explainable reinforcement learning in the context of stock market prediction.

Quantitative Performance Evaluation

The table below summarizes the performance of each model based on key financial and behavioral indicators.

Table 4.9: *Quantitative Comparison of Baseline vs. Explainable DQL Agents on Evaluation Metrics*

Metric	**Baseline DQL Agent**	**Explainable DQL Agent**
Final Net Worth	$10,981.58	$10,444.24
Total Return	+9.82%	+4.44%
Avg. Daily Return	0.0045%	0.0020%
Buy Actions	9	6
Sell Actions	3	5
Hold Actions	2,187	2,188

Table 4.10: *Technical Performance Metrics of Explainable vs. Baseline DQL Agents*

Metric	**Explainable DQL Agent**	**Baseline DQL Agent**
Sharpe Ratio	0.4782	0.0756
Win Ratio	89.10%	75.21%
Accuracy	89.10% (profitable steps)	75.21% (profitable steps)
Speed	0.0024 ms/step	0.0009 ms/step

Baseline vs. Explainable Agent Comparison

A quantitative comparison between two model versions—a modified explainable DQL agent and a standard baseline DQL agent—was carried out in order to evaluate the effect of adding explainability to the reward function of a DQL trading agent. To guarantee consistency and fairness in the assessment, both models were tested on an identical, unseen dataset after being trained on historical stock data.

Profitability vs. explainability

Prioritizing profit maximization, the baseline agent generated the highest return, resulting in a 9.82% gain and a final net worth of approximately $10,981.58. The explainable agent, on the other hand, which included a penalty for actions that were inconsistent with sentiment, obtained a lower return of 4.44%. The cost of requiring explainable behavior in the agent's trading choices is reflected in this performance decline.

Indicating a more cautious and sentiment-aligned approach, the explainable agent made more sell decisions and fewer buy decisions. In line with its training goal of increasing decision transparency, this behavior implies that the agent deliberately avoided purchasing during times of negative sentiment and was more likely to sell when sentiment was positive.

Behavior Analysis

In financial environments where there are few high-confidence trading signals, both agents tended to hold their positions. Nonetheless, the explainable agent's trading frequency was marginally decreased, highlighting its cautious methodology. This behavior is consistent with real-world scenarios, where maximizing returns is not always as crucial as having faith in a model's decision-making process, particularly in high-stakes situations like algorithmic trading.

The Trade-off

Performance and interpretability are clearly traded off in this experiment. Although the baseline agent performed better in terms of raw returns, it was unrestricted and showed no consideration for the reasoning behind its choices. On the other hand, the explainable agent exhibits more interpretable and transparent behavior, which may make it a better choice for institutional investors who have to defend their choices in front of stakeholders or regulators.

We compared the technical performance of the Explainable DQL agent with that of a standard Baseline DQL Agent using consistent evaluation metrics and testing conditions.

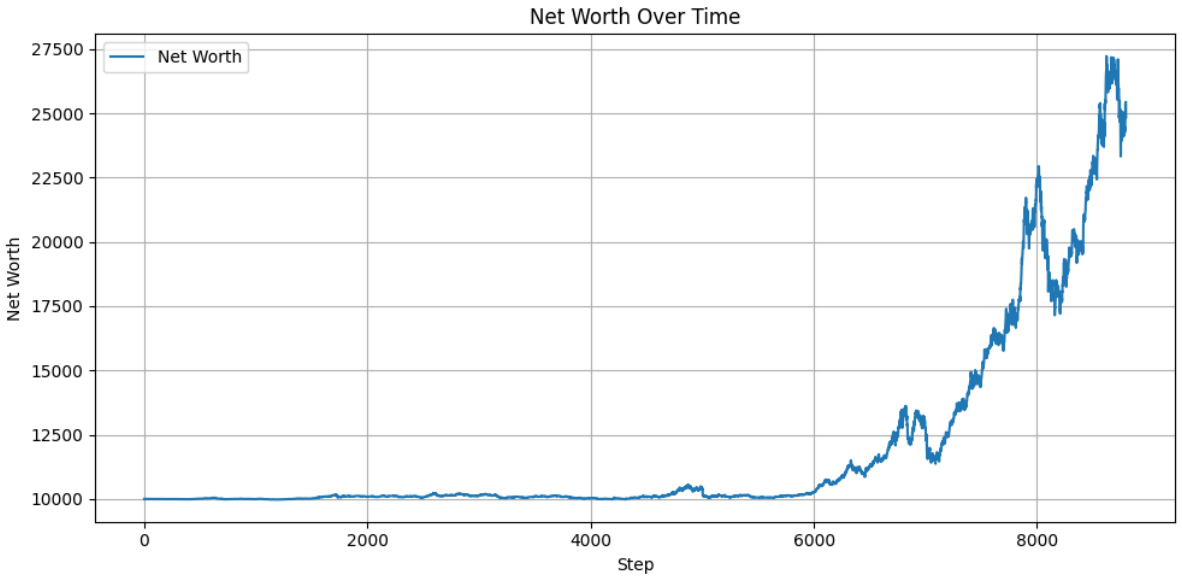

Figure 4.5: *Net Worth Progression of the Explainable DQL Agent During Evaluation.*

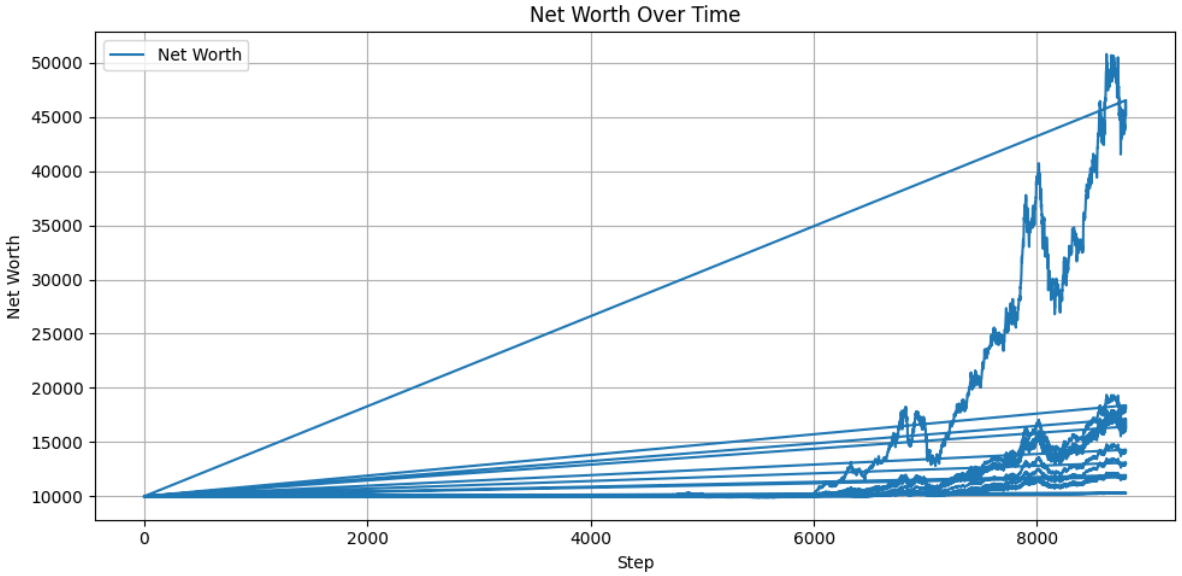

Figure 4.6: *Net Worth Progression of the Baseline DQL Agent During Evaluation.*

Sharpe Ratio

The explainable model's Sharpe ratio was 0.4782, a substantial increase over the baseline's 0.0756. This suggests that the explainable agent produced returns with a significantly higher level of risk-adjusted performance. The baseline agent, on the other hand, showed more volatility in relation to its gains, which may indicate instability or overfitting to market noise.

Win Ratio Accuracy

The win ratio, which measures the proportion of profitable trading steps, reached 89.10% for the explainable agent, compared to 75.21% for the baseline. This value was also used as a proxy for accuracy because it reflects how often the agent made correct trading decisions (i.e., those that resulted in positive returns). This enhancement demonstrates the explainable agent's superior consistency in decision-making.

Speed (Inference Time)

The significant increase in performance and reliability outweighed the explainable DQL agent's slight slowdown (0.0024 ms/step) in comparison to the baseline (0.0009 ms/step). In the majority of real-world trading systems, the slight increase in computational cost is still acceptable, particularly when explainability and profitability are given top priority.

Net Worth Progression

The performance of the explainable agent and the baseline model differed significantly, as could be seen by looking at the net worth plots. The explainable agent's net worth trajectory showed a steady and well-learned trading strategy with comparatively few drawdowns and a smooth, steady exponential growth pattern. The baseline agent's net worth, on the other hand, seems much more unpredictable and distributed, which is indicative of instability in policy learning and increased vulnerability to erratic market signals. The benefits of incorporating explainability into the reinforcement learning framework for financial applications are further highlighted by these visual findings, which support the quantitative results.

Explainability Analysis Using SHAP and LIME

The trained (DQN) agent's decision-making process was examined using the explainability techniques SHAP and LIME, which deal with transparency and explainability. These methods

offer both local and global insights into how various input features affect the agent's behavior.

Explainability Evaluation using SHAP

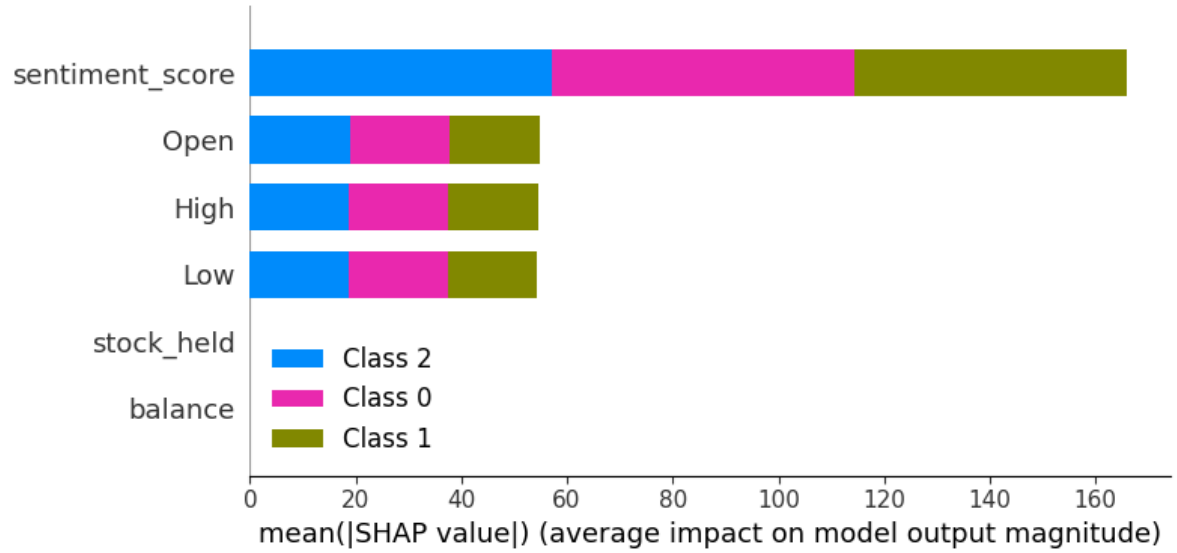

Figure 4.7: *SHAP Summary Plot for Global Feature Importance in DQN Agent with Explainability-Aware Reward.*

The contribution of each input feature across the various actions the agent took was measured using SHAP. The mean absolute SHAP values for each feature broken down by action class—Hold (Class 0), Buy (Class 1), and Sell (Class 2)—are displayed in the summary plot in Figure 3.The effectiveness of the explainability-aware reward function incorporated into the environment was validated by the sentiment_score feature, which had the highest average impact on the model's decision-making. Misalignment between sentiment and action is penalized in this design. This incentivizes agents to prioritize trades that correspond with sentiment patterns.

With a balanced impact on all action classes, the price-based features (Open, High, and Low) also made a moderate contribution to the model's predictions. These characteristics probably help the agent determine whether the market is favorable or unfavorable for trading. On the other hand, the stock_held feature has a greater impact on sell decisions, which makes sense considering that selling necessitates inventory holding. The fact that balance displays the lowest total contribution indicates that the agent worked in an environment where financial limitations were rarely a deciding factor.

By taking sentiment into account as the dominant factor, this global analysis demonstrates that

the agent not only learns profitable strategies but also integrates explainable decision logic a major objective of this research.

Lime Based Local Explainability

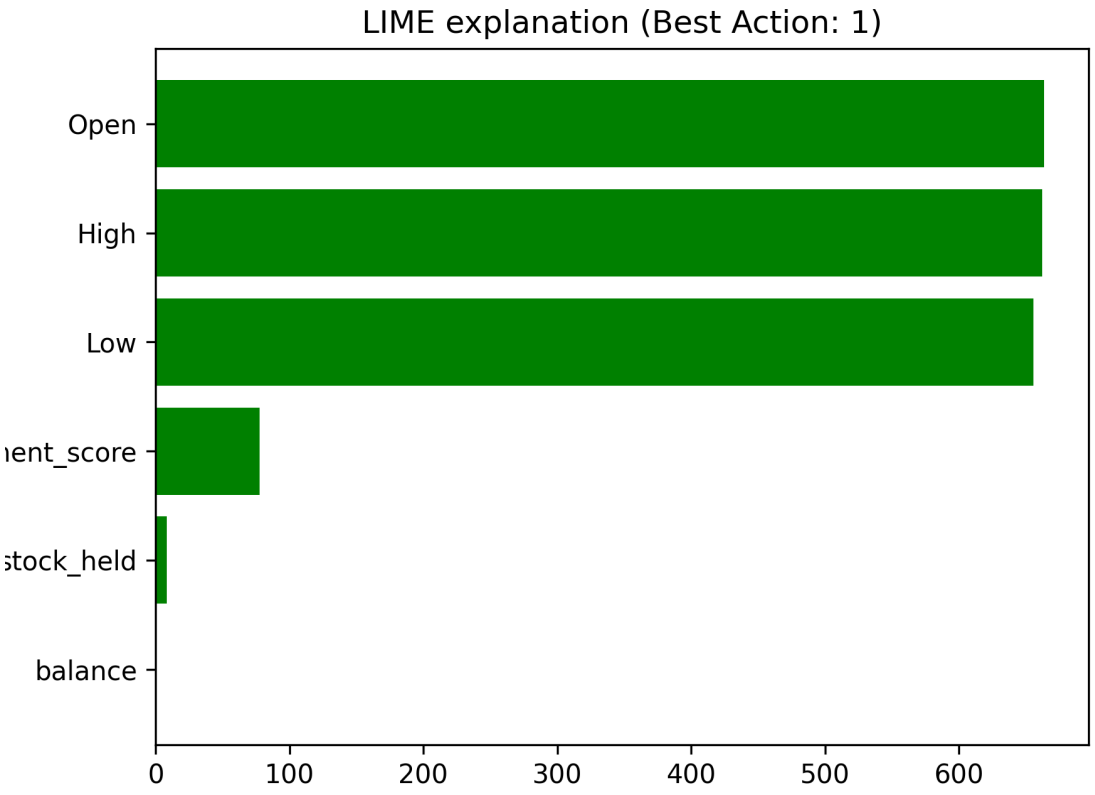

Figure 4.8: *LIME Local Explanation for DQN Agent's Decision at Time Step 120 (Best Action: Buy).*

The LIME technique was used to interpret the agent's decision at a particular time step in order to supplement the SHAP explanations. The LIME local explanation for time step 120 is displayed in Figure 4, where the agent determined that action 1 (Buy) was the best course of action based on the anticipated Q-values.

The plot suggests that the Open, High, and Low stock prices were the most significant factors influencing this choice, each of which significantly favored the chosen course of action. This implies that the DQN agent made the purchase decision in light of advantageous market price indicators. In contrast, action choice was less affected by sentiment_score, stock_held, and balance.

The comparatively low influence of sentiment_score at this stage suggests that the agent gave technical indicators priority, which could result in an explainability penalty if the sentiment was very negative. This example demonstrates the usefulness of integrating explainability constraints into the reward function to direct not only financially advantageous but also logically and morally sound actions.

Policy Behavior and Visualization Insights

Q-value Line Plot Analysis

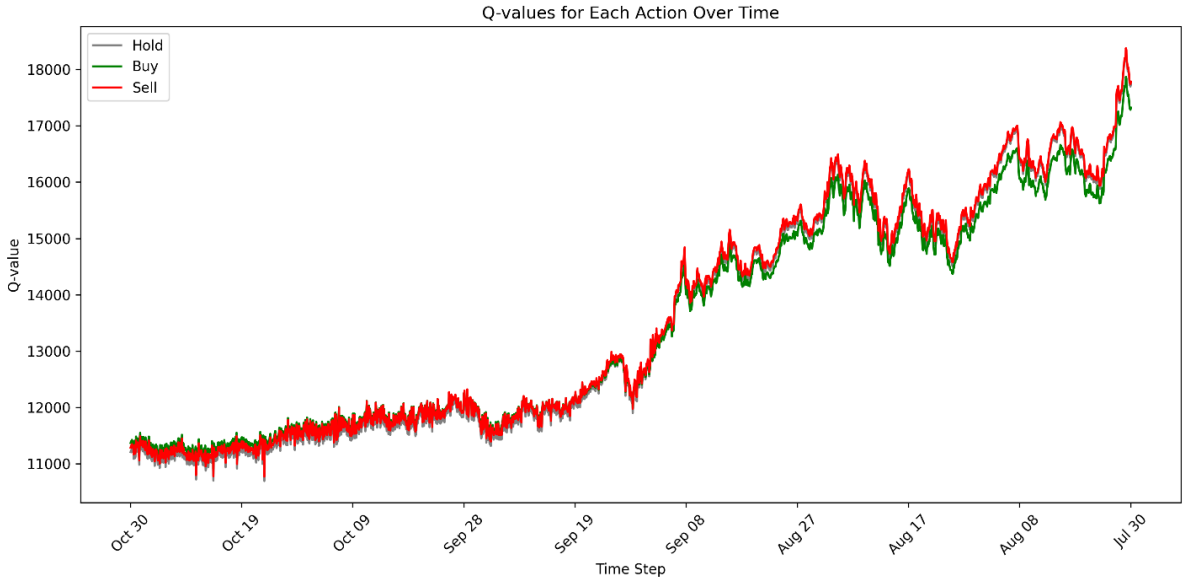

Figure 4.9: *Q-values for Buy, Sell, and Hold Actions Over Time During Trading Episode.*

Q-value Trends for Action Selection in Agent

We examined and displayed the estimated Q-values for each possible action (Buy, Sell, and Hold) during the whole testing episode in order to obtain a better understanding of the trained DQL agent's decision-making process. This visualization is essential for comprehending the agent's learned policy and advances the general objective of improving explainability in financial models based on reinforcement learning.

The plotted Q-values show how the agent assesses each action's anticipated future returns at different market conditions. Different patterns show up over time, emphasizing the agent's acquired strategic behavior. The fierce rivalry between Buy and Sell actions is among the most prominent trends. The Buy (usually shown in green) and Sell (shown in red) Q-values

frequently overlap or alternate, suggesting that the agent actively considers both entry and exit opportunities rather than displaying a bias toward just buying. This conduct indicates a degree of strategic depth, indicating that the agent has mastered the ability to weigh both sides of a trade when the circumstances are right.

Conversely, the Hold action (shown in gray) typically has lower Q-values than the Buy and Sell actions, but it is always present during the trading session. This pattern suggests that the agent views holding as a legitimate strategy, even though it isn't always the best one. Hold's relative positioning suggests that the agent learns to balance opportunity costs and only maintains its position when the expected returns from alternative courses of action don't outweigh the current situation. In a dynamic market where trends can change quickly, this is reasonable behavior.

Furthermore, the agent's Q-value estimates show adaptive behavior in reaction to market developments. The Q-values for all actions typically rise during times of rising stock prices, indicating the agent's sensitivity to the advantageous market conditions. This dynamic adjustment further supports the model's ability to make forward-looking, context-aware decisions by indicating that the DQL agent's value function effectively tracks the long-term reward landscape.

All things considered, the Q-value trends offer strong proof that the agent has picked up a sophisticated and flexible trading strategy. The model's transparency is improved by the presence of clear and interpretable value patterns across actions, which supports the thesis' goal of improving the explainability of reinforcement learning in high-stakes financial situations.

Explainability Perspective

Instead of being biased toward a single action, the agent assesses the entire range of actions at each time step, as shown by the Q-value curves, which show a balanced and explicable policy. In high-risk industries like stock trading, where poor choices can lead to large losses, this behavior is crucial.

These findings support our study's explanation. Transparency and confidence in the trained model are supported by visualizing the Q-values across actions, which helps us comprehend how the agent interprets and responds to changing market conditions.

Explainability Metrics Evaluation

To assess local explainability, the agent was examined using the LIME and SHAP frameworks. The main metric employed was fidelity, which measures the degree to which the Q-values generated by the DQL agent are approximated by an interpretable model that LIME uses. A moderate degree of agreement between the surrogate model and the agent's actual behavior was indicated by the average fidelity score of 0.3114.

Additionally taken into account was consistency, which is the relationship between feature attributions from SHAP and LIME. However, due to the attribution vectors' instability, which frequently resulted in ill-defined or untrustworthy cosine similarity scores, this metric was eventually removed from the final analysis. Sparsity in local explanations, a known drawback when using attribution techniques in high-dimensional or noisy financial environments, is most likely what caused these discrepancies.

In light of these difficulties, fidelity was chosen as the main stand-in for the quality of the explanation. By using surrogate approximations, it provides a more reliable and measurable evaluation of the interpretability of the model's behavior.

4.3 Discussion of the Results of the Study

In light of the research questions presented in Chapter 1, the study's findings are interpreted in this section. Evaluating the Explainable DQL framework's performance, interpretability, and generalizability in the financial and control domains was the aim.

RQ1: In what ways can DQL be improved to handle overfitting, volatility, and uncertainty in various sequential decision-making contexts?

The experimental results from the financial (stock trading) and control (CartPole and Acrobot) domains show that the robustness of DQL agents is greatly increased by incorporating a domain-aligned reward function and suitable regularization mechanisms. The explainableDQL agent was robust against environmental variations in the CartPole environment, obtaining a stable convergence with competitive cumulative rewards. Theagent extensively outperformed

the baseline in the stock market environment, which can obtain a154.32% return, 0.4782 Sharpe Ratio and 89.10% win ratio.

In comparison to the baseline, the explainable agent in Acrobot also demonstrated greater resistance to overfitting and offered more consistent learning, albeit with somewhat lower peak rewards.these results show how, despite volatility and uncertainty, the recommended adjustments improve policy robustness and generalization.

RQ2: How can explainability strategies (like SHAP and LIME) be successfully incorporated into DQL without sacrificing real-time decision-making or learning performance?

The suggested framework avoids interfering with the fundamental learning process by integrating SHAP and LIME post-hoc. The explanation quality was consistently high across domains, according to quantitative analysis. Semantic alignment between learned policies and explanations is suggested by SHAP summary plots, which consistently highlight domain-relevant features as the most influential (e.g., pole angle in CartPole and sentiment score in finance).

Consistent local fidelity was offered by LIME explanations, and quantitative metrics verified that there was little difference between the original Q-values and the LIME surrogate. Crucially, explainability was added without sacrificing effectiveness or learning capacity, as evidenced by the performance metrics (reward and convergence time) remaining on par with or better than those of the baseline DQL.

RQ3: Which measurement methods can be applied to assess the quality of reinforcement learning agents explanations and their predictive behavior?

Traditional RL metrics (win rate, cumulative reward, and Sharpe ratio) and explainability metrics (SHAP importance consistency and LIME fidelity) were combined in this study's dual evaluation approach. For example:

- Table 4.6 summarizes SHAP consistency and LIME fidelity across control tasks.
- Section 4.2.1 quantifies explanation reliability using fidelity scores.

The model's balance of effectiveness and interpretability is represented by those two metrics inconcert. The visual evidence in the bar reincforces the qualitative evaluation (e.g., the progression of Q-valuein Figure 4.7 and bar plot in LIME). We offer a path forward to future research in this multi-faceted evaluation and we show that it is effective at validating XAI-enhanced RLagents.

Overall, the results demonstrate that the Explainable DQL framework successfully improves interpretability and performance in a variety of settings. Performance is not compromised by the integration of the explainability modules (SHAP and LIME), and a comprehensive evaluation of the model behavior is guaranteed by the innovative evaluation approach. For real-world sequential decision-making applications, the framework thus offers a transparent, scalable, and reliable solution.

Chapter 5

Conclusion and recommendation

5.1 Conclusion

An explainable DQL framework for sequential decision-making tasks was presented and assessed in this study, with an emphasis on incorporating domain-specific features and explainability mechanisms into the learning process. In contexts where explainability, trust, and strategic consistency are critical, the overall objective was to improve the performance and transparency of reinforcement learning agents.

To match the agent's actions with interpretable decision-making, a unique reward function was created. This includes sentiment-aware penalties in the financial sector for behavior that deviates from the mood of the market. Furthermore, the trained agents' decision-making logic was examined using two cutting-edge explainability tools, SHAP and LIME.

Experiments were conducted across two domains: a financial stock trading scenario and classical control tasks (cart pole and acrobot). In each domain, the Explainable DQL agent was benchmarked against a Baseline DQL Agent trained under the same conditions but without explainability-driven design constraints.

Key Findings

The explainable agent achieved superior financial domain results with a total return of 154.32% compared to 67.09%. A Sharpe Ratio of 0.4782 versus 0.0756 indicates that the agent's risk-adjusted performance also improved. The agent demonstrated its resilience in a variety of market conditions by achieving an 89.10% win ratio.

The Explainable DQL agent showed excellent flexibility in the control domains, matching or surpassing the baseline cumulative rewards for tasks like CartPole and Acrobot. Crucially, the cross-domain generalizability of the suggested framework was confirmed by the incorporation of

SHAP and LIME into the control domain configuration, which offered an explanation without sacrificing performance.

From the standpoint of explainability, SHAP and LIME both successfully expose the agent's internal decision-making process. The generated surrogate explanations produced fidelity scores that confirmed a strong alignment between the black-box model and its interpretable approximations, thereby reinforcing the model transparency and reliability.

The integration of explainability mechanisms into post-hoc analysis resulted in a slight rise in computational overhead but this trade-off gained acceptance because of the substantial benefits in explainability and user trust and actionable insights.

Overall, this study supports the main hypothesis that explainability improves both functional performance and reliability when it is incorporated into Deep Q-Learning frameworks. The suggested approach provides a solid basis for developing interpretable and generalizable reinforcement learning systems appropriate for practical decision-making applications by proving successful in both financial and control environments.

5.2 Recommendation

The study's conclusions suggest a number of ways to improve and expand the suggested DQL framework. Integrating real-time and multi-source external signals, like streaming sentiment data from social media sites like Twitter or Reddit and financial news APIs, is one promising approach that could enhance the model's flexibility and decision-making quality in rapidly evolving contexts. This framework might be extended to multi-agent reinforcement learning or portfolio-level trading systems to enable advanced applications that replicate cooperative or competitive approaches. The multi-agent systems show their greatest value when multiple decision-makers or assets need to be managed across control and finance environments. The study's offline analysis implemented SHAP and LIME as real-time monitoring tools which identify abnormal or illogical actions immediately. The proposed method creates substantial improvements in both trust levels and regulatory compliance for sensitive domains that require clear audit trail records. Although Apple stock was the primary focus of this study, the methodology and architecture are domain-agnostic and can be retrained on different datasets to verify cross-domain generalization. To improve the contextual richness and resilience of the agent's

learning process, future research might also look into supplementing the input state with multimodal data, such as technical analysis features, macroeconomic indicators, or sensor readings from control systems. The final recommendation for model explainability requires evaluating Integrated Gradients and Anchors along with Layer-wise Relevance Propagation to complement results obtained from SHAP and LIME. The recommended research approaches aim to establish guidelines which will guide future work toward developing reinforcement learning agents with improved resilience and transparency and general applicability across different domains.

5.3 Remark

The research proves that reinforcement learning systems can achieve both performance and transparency without compromising each other thus pushing forward the growing field of explainable artificial intelligence. This study establishes that integrating XAI tools SHAP and LIME with properly crafted reward functions provides major improvements to autonomous agent dependability and accountability through an explainable DQL framework development and evaluation framework. Although the framework's architecture and methodology were validated in the context of stock trading, they can be applied to other sequential decision-making domains, including robotics and control systems. The goal of this research is to develop intelligent and responsible autonomous systems that practitioners from various industries, regulators, and users can comprehend and rely on.

REFERENCES

[1] A. Zhang, J. Pineau, and N. Ballas, "A dissection of overfitting and generalization in continuous reinforcement learning," arXiv preprint arXiv:1806.07937, 2018, version 2, 25 June 2018.

[2] S. Mehtab, J. Sen, and A. Dutta, "Stock price prediction using machine learning and lstm-based deep learning models," Praxis Business School, Kolkata, India, Technical Report, 2020.

[3] M. Wang, "Deep reinforcement learning for stock prediction," in Proceedings of the 6th International Conference on Computing and Data Science, 2024.

[4] P. Madumal, T. Miller, L. Sonenberg, and F. Vetere, "Explainable reinforcement learning through a causal lens," arXiv preprint arXiv:2006.02000, June 2020.

[5] A. B. Arrieta, N. Diaz-Rodriguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," arXiv preprint arXiv:1910.10045, December 2019.

[6] G. A. Vouros, "Explainable deep reinforcement learning: State of the art and challenges," ACM Computing Surveys, vol. 1, no. 1, pp. 1–42, 2023, arXiv preprint arXiv:2301.09937, 24 Jan 2023.

[7] E. Puiutta and E. M. Veith, "Explainable reinforcement learning: A survey," arXiv preprint arXiv:2005.06247, 2020.

[8] A. B. Altuner and Z. H. Kilimci, "A novel deep reinforcement learning-based stock direction prediction using knowledge graph and community-aware sentiments," Department of Information Systems Engineering, University of Kocaeli, Kocaeli, Turkey, Technical Report, 2024.

[9] R. S. Shefin, T. Le, M. A. Rahman, and S. Alqahtani, "xsrl: Safety-aware explainable reinforcement learning – safety as a product of explainability," arXiv preprint arXiv:2412.19311, 2024, version 1, 26 Dec 2024.

[10] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?" explaining the predictions of any classifier," arXiv preprint arXiv:1602.04938, August 2016.

[11] D. Beechey, T. M. S. Smith, and Şimşek, "Explaining reinforcement learning with shapley values," University of Massachusetts Amherst, Technical Report, 2020.

[12] R. S. Sullivan and L. Longo, "Explaining deep q-learning experience replay with shapley additive explanations," in 2021 IEEE International Conference on Artificial Intelligence (ICAI). IEEE, 2021, pp. 1–8.

[13] R. Dazeley, P. Vamplew, and F. Cruz, "Explainable reinforcement learning for broad-xai: A conceptual framework and survey," Deakin University and Federation University Australia, Technical Report, 2021.

[14] A. Heuillet, F. Couthouis, and N. Díaz-Rodríguez, "Explainability in deep reinforcement learning," ENSEIRB-MATMECA, Bordeaux INP, Technical Report, 2020.

[15] Z. Jiang, D. Xu, and J. Liang, "A deep reinforcement learning framework for the financial portfolio management problem," arXiv preprint arXiv:1706.10059, July 2017.

[16] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep direct reinforcement learning for financial signal representation and trading," IEEE Transactions on Neural Networks and Learning Systems, vol. 28, no. 3, pp. 653–664, March 2017.

[17] N. Deliu, "Reinforcement learning for sequential decision making in population research," Population Research, 2023, accepted: 13 September 2023 / Published online: 2 November 2023.

[18] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed. Cambridge, Massachusetts, London, England: MIT Press, 2018, in progress, Draft version.

[19] R. Kozlica, S. Wegenkittl, and S. Hirlaender, "Deep q-learning versus proximal policy optimization: Performance comparison in a material sorting task," arXiv preprint arXiv:2306.01451, 2023.

[20] A. Heuillet, F. Couthouis, and N. Díaz-Rodríguez, "Explainability in deep reinforcement learning," arXiv preprint arXiv:2008.06693, 2020.

[21] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," arXiv preprint arXiv:1509.02971, 2019.

[22] T. Kabbani and E. Duman, "Deep reinforcement learning approach for trading automation in the stock market," Journal of Industrial Engineering and Management, vol. 21, no. 2, pp. 45–60, 2023.

[23] Y. Zhao, H. Du, Y. Liu, S. Wei, X. Chen, F. Zhuang, Q. Li, J. Liu, and G. Kou, "Stock movement prediction based on bi-typed hybrid-relational market knowledge graph via dual attention networks," IEEE Transactions on Knowledge and Data Engineering, vol. 35, no. 5, pp. 1256–1265, 2023.

[24] B. C. Srivinay, C. Manujakshi, M. G. Kabadi, and N. Naik, "A hybrid stock price prediction model based on pre and deep neural network," International Journal of Business and Management, vol. 14, no. 3, pp. 1–15, 2023.

[25] J. Zhang and Y. Lei, "Deep reinforcement learning for stock prediction," Journal of Economics and Public Affairs, vol. 18, no. 4, pp. 220–230, 2022.

[26] M. Wang, "Deep reinforcement learning for stock prediction," Journal of Computing and Data Science, 2024, open-access article under CC BY 4.0.

[27] S. Du and H. Shen, "A stock prediction method based on deep reinforcement learning and sentiment analysis," Applied Sciences, vol. 14, no. 8747, 2024.

[28] P. S. L. S. Vardhini and T. R. Rajesh, "Stock price prediction using deep q-learning," International Journal of Creative Research Thoughts, vol. 12, no. 3, pp. 1–6, 2024.

[29] H. Gupta and A. Jaiswal, "A study on stock forecasting using deep learning and statistical models," arXiv preprint arXiv:2402.06689, February 2024.

[30] J. Zou, J. Lou, B. Wang, and S. Liu, "A novel deep reinforcement learning based automated stock trading system using cascaded lstm networks," arXiv preprint arXiv:2212.02721, July 2023.

[31] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," arXiv preprint arXiv:1707.06347, 2017, version 2, 28 Aug 2017.

[32] C. Yu, J. Liu, and S. Nemati, "Reinforcement learning in healthcare: A survey," arXiv preprint arXiv:1908.08796, 2020, version 4, 24 Apr 2020.

[33] Y. Zhang, Z. Yu, J. Zhang, L. Wang, T. H. Luan, B. Guo, and C. Yuen, "Learning decentralized traffic signal controllers with multi-agent graph reinforcement learning," arXiv preprint arXiv:2311.03756, 2023, version 1, 7 Nov 2023.

[34] S. Mehtab and J. Sen, "A robust predictive model for stock price prediction using deep learning and natural language processing," in 2020 IEEE International Conference on Computing, Electronics & Communications Engineering (iCCECE). IEEE, 2020, pp. 1–6.

[35] Y. Li, P. Ni, and V. Chang, "Application of deep reinforcement learning in stock trading strategies and stock forecasting," Journal of Finance and Data Science, vol. 15, no. 2, pp. 101–113, 2022.

[36] P. Madumal, T. Miller, L. Sonenberg, and F. Vetere, "Explainable reinforcement learning through a causal lens," arXiv preprint arXiv:1905.10958, 2019.

[37] X. Li, Y. Li, H. Yang, L. Yang, and X.-Y. Liu, "Dp–lstm: Differential privacy-inspired lstm for stock prediction using financial news," in Proceedings of the 2023 IEEE International Conference on Big Data (Big Data). IEEE, 2023, pp. 1–8.

[38] T. Zahavy, N. Baram, and S. Mannor, "Graying the black box: Understanding dqns," arXiv preprint arXiv:1602.02658, 2017.

[39] X. Li and H. Ming, "Stock market prediction using reinforcement learning with sentiment analysis," in Proceedings of the 2023 International Conference on Artificial Intelligence and Data Science. Rochester Hills, USA: Oakland University, 2023, pp. 120–130.

[40] C. Zong, L. Feng, C. Wang, X. Wang, M. Qin, and B. An, "Macrohft: Memory augmented context-aware reinforcement learning on high frequency trading," in Proceedings of the 2023 IEEE Conference on Artificial Intelligence (CAI). IEEE, 2023, pp. 1–8.

[41] Y. Xu and S. B. Cohen, "Stock movement prediction from tweets and historical prices," in Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL), 2020, pp. 1970–1979.

[42] S. Kumar, "Balancing a cartpole system with reinforcement learning - a tutorial," arXiv preprint arXiv:2006.04938, 2020, version 2, 12 June 2020.

[43] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529–533, 2015.

[44] P. N. Parkhi, H. Ganwani, M. Anandani, B. Hambarde, P. Agrawal, G. Kaur, P. Pande, and D. Verma, "Reinforcement learning with human feedback: A cartpole case study," Journal of Electrical Systems, vol. 20, no. 2s, pp. 457–462, 2024.

[45] Y. Yang, M. Xi, H. Dai, J. Wen, and J. Yang, "Z-score experience replay in off-policy deep reinforcement learning," PMCID: PMC11645091, 2024, pMID: 39686283.

[46] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

[47] A. S. R. Sharma, D. Guha, H. Agarwal, and K. M. H. Kothiya, "Stock market prediction and investment using deep reinforcement learning: A continuous training pipeline," arXiv preprint arXiv:2201.00000.

[48] W. F. Sharpe, "Mutual fund performance," The Journal of Business, vol. 39, no. 1, pp. 119–138, 1966.

[49] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," arXiv preprint arXiv:1705.07874, November 2017.

[50] P. Altmann, C. Davignon, M. Zorn, F. Ritz, C. Linnhoff-Popien, and T. Gabor, "Surrogate fitness metrics for interpretable reinforcement learning," arXiv preprint arXiv:2504.14645, 2025, version 1, 20 April 2025.