Faculty of Natural and Mathematical Sciences
Department of Informatics

Bush House, King's College
London, Strand Campus, 30
Aldwych, London WC2B 4BG
Telephone 020 7848 *2145*
Fax 020 7848 2851

7CCSMPRJ

Individual Project Submission 2024/25

Name: Gleb Romantchik

Student Number: 24104079

Degree Programme: MSc Computational Finance

Project Title: Mining for Patterns with Deep Reinforcement Learning in Commodity Markets

Supervisor: Dr. Peter McBurney

Word count: 8,971

RELEASE OF PROJECT

Following the submission of your project, the Department would like to make it publicly available via the library electronic resources. You will retain copyright of the project.

☑ I **agree** to the release of my project

☐ I **do not** agree to the release of my project

Signature: Gleb H. Romantchik

Date: 5/08/2025

Abstract

Once-profitable commodity roll strategies that generated Sharpe ratios of 4.39 have collapsed to near-zero returns due to algorithmic competition, yet $100+ billion in monthly roll flows persist. This creates an ideal testing ground for AI-based pattern recognition.

This project investigates whether AI methods, specifically Deep Reinforcement Learning, can identify and exploit trading opportunities in commodity futures roll periods using a Double Deep Q-Network (DDQN) combined with Long Short-Term Memory (LSTM) networks. Using 15 years of Bloomberg data (2010-2025) across across energy, metals, softs and livestock futures, this work tests whether Deep Reinforcement Learning can detect patterns invisible to traditional strategies.

The methodology combines practical trading experience with systematic analysis of various reinforcement learning applications in financial markets. Feature selection and data is informed by both quantitative analysis and qualitative insights from experienced commodity traders who previously operated successful roll strategies.

Results demonstrate that while the DDQN algorithm achieves statistically significant pattern recognition capabilities - generating 50.6% average win rates compared to 0% for traditional Goldman Roll strategies in energy markets - transaction cost erosion negates net profitability across all tested commodities. The algorithm's superior performance in Gold and Live Cattle markets suggests that AI methods can detect residual roll patterns where mechanical strategies fail entirely.

This work provides the first rigorous evaluation of deep reinforcement learning for commodity roll trading, establishing realistic benchmarks through statistical validation and comprehensive transaction cost modeling - friction typically omitted in academic literature but critical for practical algorithmic trading applications.

Acknowledgements

I would like to thank my supervisor, Dr. Peter McBurney, for his guidance, patience, and expertise throughout this research project. His insights and encouragement were instrumental in shaping this work.

Special thanks are due to the experienced commodity traders who generously shared their market knowledge and practical insights into roll trading strategies. Their real-world expertise provided crucial context that bridged the gap between academic theory and market reality.

I would also like to thank my family and friends for their unwavering support and encouragement throughout my studies. Their patience during the countless hours spent coding, analysing data, and writing was deeply appreciated. Any errors or omissions in this work remain entirely my own responsibility.

Contents

List of Figures

List of Tables

Nomenclature

Mathematical Symbols

Symbol	Definition	Units/Dimensions
γ	Discount factor	Dimensionless
α	Learning rate	Dimensionless
ϵ	Exploration parameter	Dimensionless
pos_t	Current position at time t	Contracts
pos_{max}	Maximum position size	Contracts (default: 8)
K	Initial capital	Dollars ($)
R_t	Reward at time t	Dollars ($)
G_t	Expected cumulative reward	Dollars ($)
s_t	State at time t	Vector
a_t	Action at time t	Discrete set
$\mathcal{A}$	Action space	Dimensionless
$\mathcal{D}$	Experience replay buffer	Dimensionless
θ	Online network parameters	Dimensionless
θ^-	Target network parameters	Dimensionless

Acronyms

Acronym	Definition
AI	Artificial Intelligence
CNN	Convolutional Neural Network
DDQN	Double Deep Q-Network
DL	Deep Learning
DRL	Deep Reinforcement Learning
DQN	Deep Q-Network
GSCI	Goldman Sachs Commodity Index
LSTM	Long Short-Term Memory
MDP	Markov Decision Process
ML	Machine Learning
OHLC	Open-High-Low-Close (prices)
OTC	Over-The-Counter
P&L	Profit and Loss
RL	Reinforcement Learning
TD	Temporal Difference
TD3	Twin-Delayed Deterministic Policy Gradient
WTI	West Texas Intermediate

Key Terms

Term	Definition
Backtesting	Testing a strategy on historical data
Bid-ask slippage	Difference between expected and actual trade price
Calendar spread	Price difference between contract months
Commodity roll	Process of moving futures positions to next contract
Drawdown	Decline from peak capital
Experience replay	Storing past experiences for training
Front-running	Trading ahead of known future orders
Goldman Roll	GSCI index rebalancing strategy
Markov Decision Process	Mathematical framework for RL
Open interest	Number of outstanding contracts
Sharpe ratio	Risk-adjusted return measure

1. Background

1.1 Introduction, Research Motivation and Objectives

Traditional discretionary trading strategies face increasing pressure from algorithmic competition and market efficiency improvements. Can AI methods, particularly reinforcement learning (RL) algorithms, identify and exploit trading opportunities where discretionary methods have become ineffective? This study is motivated by three key factors: direct trading experience of a declining strategy, the success of machine learning in pattern recognition tasks and the continued existence of systematic flows that should, at least theoretically, be exploitable.

This work's structure is as follows: after this background section a comprehensive literature review will examine existing applications of RL to financial markets, compare different algorithmic approaches, and establish best practices for this work's approach. The methodology section will detail data acquisition, feature engineering, algorithm implementation, and validation procedures. Finally, results will be evaluated using appropriate performance metrics, with particular attention to returns and practical implementation considerations.

1.2 Commodity Futures Roll Trading

The primary motivation for this research stems from direct trading experience with strategies targeting Goldman Sachs Commodity Index (GSCI) rebalancing flows. Launched in 1991, it became one of the most influential commodity benchmarks, with assets under management reaching over $100 billion at its peak. The GSCI methodology requires monthly 'rolling' of near-term futures contracts into longer-dated contracts to maintain continuous exposure, creating predictable and substantial trading flows. Traditional roll trading strategies focused on identifying these systematic flows through analysis of open interest changes, volume patterns, and commodities' calendar spread movements.

These strategies, once highly profitable for discretionary traders, have seen their profitability reduced due to increased algorithmic participation. As will be expanded upon in the literature review, Irwin et al. (2022) document a dramatic decline in exploitable alpha from these roll strategies, falling by over 80% between the early 2000s and late 2010s.

However, the underlying market mechanics that created these opportunities continue to occur every month across global commodity markets. This presents a compelling research question: can AI methods detect and exploit trading opportunities in this environment where human pattern recognition has become insufficient?

The success of artificial intelligence in other domains, particularly the recent advances in large language models, demonstrates the power of machine learning to discover and exploit patterns. While machine learning applications in finance are extensive, the specific application of reinforcement learning to commodity roll trading remains underexplored in academic literature. This gap between persistent market flows and declining human trading success provides an ideal test case for examining whether machine learning can detect patterns beyond human analytical capabilities.

2. Literature Review

This literature review traces the evolution of machine learning from neural network origins to the foundations that underpin modern RL applications in markets. This review will provide context on the evolution of machine learning (ML) to a level surpassing human performance.

2.1 Machine Learning and Neural Network Origins

McCulloch and Pitts' seminal (1943) work introduced the first mathematical model of artificial neural networks. Their work, while purely theoretical and lacking an actual learning mechanism, provided the computational basis for artificial intelligence - see Figure 2.1. Building upon this theory, Rosenblatt (1958) introduced the first algorithmically trainable neural network. The 'perceptron' represented a significant advancement by designing a learning mechanism that could adjust 'weights' based on training data. A weight is a numerical parameter that determines the strength of influence each input feature has on the model's final decision (for example, 'buy' or 'sell'). Rosenblatt's hardware implementation, the Mark I Perceptron, demonstrated practical learning from data for the first time. However, it could only solve simple problems (like separating circles from squares).

The breakthrough that revitalized neural network research came with Rumelhart et al.'s (1986) paper "Learning Representations by Back-Propagating Errors". To understand their advancement, it is essential to first clarify the key concepts involved. A layer in a neural network is a group of interconnected artificial neurons that all process information at the same level, with data flowing from one layer to the next like steps in a factory assembly line. 'Deep learning' uses neural networks with many layers to automatically learn complex patterns from data, similar to how the human brain processes information through multiple levels of understanding. The primary challenge was the 'credit assignment problem': when a multi-layer network makes a mistake, it was unclear which specific layers or neurons were responsible for the error and how much each should be adjusted to fix it. Rumelhart et al.'s backpropagation algorithm addressed this challenge by teaching neural networks to work backwards from wrong answers to figure out which parts of the network made mistakes and how to fix them.

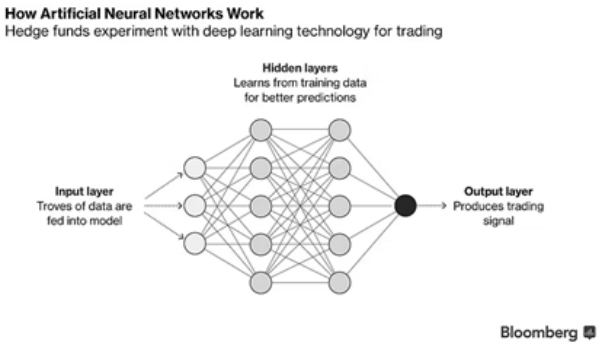

Figure 2.1: Neural network illustration (Lee, 2025).

2.2 The Emergence of Reinforcement Learning

Neural networks and deep learning revolutionised 'supervised learning' - where algorithms learn from labeled examples - many real-world problems require decision-making in an unfamiliar environment. This limitation of supervised learning led to the development of 'reinforcement learning' (RL): algorithms constructed as agents, learning optimal behaviors through trial-and-error processes, receiving rewards or penalties based on their actions and thereby 'learning' what the most optimal action is to take.

Sutton (1988) established the theoretical foundation for reinforcement learning through his Temporal Difference (TD) algorithm, which learns by comparing predicted outcomes with actual results and adjusting future predictions based on this error signal. Building upon this principle, Watkins (1989) introduced 'Q-learning', a foundational value-based algorithm where agents learn to estimate Q-values representing the expected utility of each action in given situations. Q-learning operates through iterative cycles: agents observe a 'state space' (all possible situations in an environment), select actions from an 'action space' (all possible moves it can make), receive performance-based rewards, and update their Q-value estimates using Sutton's TD error corrections. This elegant combination of state-action evaluation with trial-and-error learning establishes the conceptual framework that underpins modern reinforcement learning approaches.

2.3 The Deep Reinforcement Learning Revolution

The modern deep learning era was catalysed by the landmark paper "Long Short-Term Memory" (LSTM) by Hochreiter and Schmidhuber (1997). LSTM could capture long-term dependencies within sequential data. Their architecture, featuring memory cells and gating mechanisms, enabled stable learning over extended time horizons and mitigated the vanishing gradient problem. 'Gradients' are the learning signals that tell each part of a neural network how much to change the weights to improve performance. Gradients 'vanish' when they become too weak to reach the early layers of very deep networks, so they stop learning effectively. LSTM solved benchmark tests with minimal time lags in excess of 1,000 discrete time steps - problems that prior recurrent network algorithms could not solve - achieving more successful runs and faster learning.

Deep learning was further advanced by Krizhevsky et al.'s (2012) paper "ImageNet Classification with Deep Convolutional Neural Networks," commonly known as AlexNet. Their work achieved a revolutionary breakthrough in pattern recognition, reducing error rates from 26.2% to 15.3% through an 8-layer 'convolutional neural network' (CNN) that learned features directly from raw pixels rather than relying on hand-crafted feature engineering. Mnih et al.'s (2013) paper "Playing Atari with Deep Reinforcement Learning," combined CNNs with Q-Learning, introducing Deep Q-Networks (DQN). Combining the superior pattern recognition of CNNs with the decision-making of Q-learning. DQN's success on Atari games marked the beginning of the 'deep reinforcement learning' (DRL) era.

The main limitation of DQN is 'overestimation bias', which occurs when the algorithm consistently overestimates how good its actions will be, leading to overly optimistic Q-value predictions and suboptimal decision-making. Van Hasselt et al. (2016) addressed this with Double DQN (DDQN) by decoupling action selection from action evaluation using two separate networks: one 'Online Network' which selects the best action and a 'Target Network' which evaluates the Q-value of that selected action - essentially a second opinion. The double-network approach demonstrated clear advantages in gaming environments like Atari Pong, achieving an 82% vs 62% win rate compared to DQN (Nivedha et al., 2024).

However, results in controlled, gaming environments with deterministic reward structures and stationary dynamics may not translate directly to financial markets which exhibit non-stationary patterns, regime changes, and adversarial behavior from other market participants.

2.4 AI Surpassing Human Performance

Silver et al.'s (2016) AlphaGo algorithm achieved superhuman performance in the ancient game of Go. AlphaGo combined CNNs for position evaluation and move prediction, with Monte Carlo Tree Search for strategic planning and demonstrated that AI systems could exceed human expertise in complex strategic reasoning. AlphaGo's 60-0 online winning streak and estimated ELO rating of 3600+ compared to humans' 3000-3200 demonstrate not just victory, but complete dominance. The evolution continued with the AlphaZero and MuZero series, which eliminated the need for human domain knowledge and environment models. MuZero's ability to learn both environmental dynamics and optimal policies purely from experience shows that it can learn how to win any game or solve any problem without being told the rules - it figures out both how the system works and the best strategy purely by trial and error (Silver et al., 2020).

In the markets, AI methods have demonstrated superior performance over traditional models. Gu, Kelly, and Xiu's (2020) study demonstrates that ML can improve portfolio performance by capturing nonlinear relationships between variables missed by traditional linear methods. Testing 13 different approaches on nearly 30,000 stocks over 60 years with 94 characteristics, they find neural network-based strategies deliver impressive results compared to the OLS-3 benchmark model: monthly out-of-sample R^2 of 0.40% compared to 0.16%. Their work reveals that nonlinear methods excel by detecting complex interactions between these variables that linear models cannot capture.

Several trading firms have embraced AI methods in their investment strategies. Renaissance Technologies stands out as one of the most successful hedge funds ever, delivering annualised gross returns of over 60% for over two decades driven by advanced machine learning algorithms (Zuckerman, 2019). AQR Capital Management, managing over $130 billion in assets, now uses machine learning for about 20% of the signals in its flagship multi-strategy fund and has launched two new strategies entirely built on AI (Lee, 2025).

However, AQR's own Israel, Kelly, and Moskowitz (2020) argue in their paper "Can Machines 'Learn' Finance?" that using ML for return prediction suffers from extremely low signal-to-noise ratios and evolving market dynamics that erode predictive signals through arbitrage. Complex nonlinear models can deliver 50-100% performance improvements over traditional linear approaches, however, this depends on using high-quality, economically meaningful signals. Adding just two noise variables can cut performance in half. Complexity without economic foundation leads to inferior results rather than improved outcomes (Portfolio Solutions Group, 2024).

2.5 The Goldman Roll

The Goldman Roll refers to the predictable monthly rolling activity of the GSCI, where index funds systematically sell expiring nearby futures contracts and purchase longer-dated deferred contracts from the 5th to 9th business day of each month to maintain commodity exposure. Simple front-running strategies exploiting this predictable price pressure could achieve extraordinary returns, with Sharpe ratios reaching 4.39 during the 2000-2010 period (Mou, 2011).

However, Irwin et al. (2022) reveal that order flow costs peaked during 2004-2011 at an average of 3.19% annually, before declining dramatically to just 0.46% for 2012-2019, representing an over 80% reduction in exploitable alpha. This is attributed to the increased market awareness of the rolling patterns that attracted additional arbitrage capital, creating an environment where only more sophisticated analytical approaches may detect residual patterns.

2.6 Theoretical Promise versus Reality

While several studies have explored RL applications across various asset classes, none have specifically addressed commodity futures roll strategies, with index rebalancing studies providing the closest parallel. Chavan et al (2021) demonstrate that RL techniques in index rebalancing can achieve superior recording higher returns and Sharpe ratios compared to passive indexing .

Mussema et al.'s (2025) study on the Ethiopian Commodity Exchange showed DDQN significantly outperforming DQN, achieving a suspiciously high Sharpe ratio of 11.64 likely by omitting transaction costs and slippage. Similarly Gao's (2024) work claims DDQN superiority over standard DQN, yet lacks rigorous evaluation metrics and relies on single-stock testing without proper baselines. Brim et al. (2022) demonstrate DDQN's potential during volatile market conditions with their CNN-enhanced model generating 13.2% returns against the S&P 500's decline over the same period, yet exclude transaction costs, bid-ask spreads, and slippage. These are all factors that can reduce or eliminate the alpha these algorithms claim to generate.

Notable exceptions exist: Kabbani and Duman (2022) incorporate transaction costs and liquidity constraints into their Twin Delayed Deep Deterministic Policy Gradient (TD3) stock trading model, achieving a more realistic 2.68 Sharpe ratio. While their continuous action space approach offers theoretical advantages over DDQN's discrete actions, it requires higher computational resources by utilising six neural networks. Bagheri (2024) uses logarithmic utility functions, achieving 22% higher returns compared to linear utility functions across multiple asset classes. Logarithmic utility captures diminishing marginal utility, encouraging behavior that avoids capital-destroying bets .

2.7 Conclusion

This review demonstrates how ML has evolved from the simple perceptron to DRL systems capable of superhuman performance. However, there still exists a significant disconnect between academic promise and practical reality in applying AI methods to commodities markets. While studies consistently report impressive results, the exclusion of realistic transaction costs, slippage, and bid-ask spreads renders results commercially meaningless.

As the Goldman Roll strategy's decline demonstrates, patterns get arbitraged away as markets adapt. Sustainable trading success requires increasingly sophisticated approaches that can identify and exploit ever-more-subtle market inefficiencies.The predictable and persistent nature of GSCI roll flows provides an ideal test case for examining the limits of algorithmic pattern recognition.

3. Methodology

This section will cover the development of the DRL approach for the Goldman Roll strategy. Three critical research gaps must be addressed: market friction costs, testing across diverse conditions and comparison against benchmark strategies, ensuring that results reflect realism rather than unrealistic assumptions.

3.1 RL Architecture Design

3.1.1 Pattern Recognition

Among the various neural network architectures available for prediction, LSTM networks are most suitable for this study due to their design for sequential data processing. While CNNs excel at identifying spatial patterns in grid-structured data i.e. image recognition (LeCun et al., 1998), financial data lacks such spatial structure - variables are arbitrarily ordered with no inherent neighborhood relationships that would benefit from convolution operations. In contrast, LSTMs are specifically designed to capture patterns that unfold over time such as futures rolling - where volume migration typically precedes open interest shifts, which in turn precede price moves. A CNN would inappropriately attempt to find spatial patterns in what is fundamentally a time-ordered process. Given this temporal modeling advantage, LSTMs demonstrate strong performance in trading pattern detection applications. Moghar et al. (2020) and Phuoc et al. (2024) apply 4-layer LSTM networks to predict stock prices. Results showed 93% accuracy in short-term price forecasting.

3.1.2 Decision Making

While the literature review identifies TD3 as the most comprehensive algorithm due to its continuous action space capabilities - offering superior flexibility for position allocation and market interpretation - it is better suited for multi-asset portfolio construction rather than the single-commodity analysis approach employed in this study. TD3's six-network architecture, while powerful for complex portfolio optimisation, introduces unnecessary computational overhead and potential failure modes when applied to individual commodity roll strategies.

Given these considerations DDQN emerges as the optimal choice for this study's specific scope and requirements. DDQN's simpler two-network structure ensures predictable training behavior and robust policy learning. The discrete action space adequately accommodates the practical needs of roll trading by way of position changes (0%, 25%, 50%, -25%, -50%) and entry timing decisions (Day 5, Day 6, Day 7). Crucially, DDQN's interpretable Q-values provide clear decision rationale for each action, enabling effective strategy validation and risk oversight - essential when deploying algorithmic trading strategies in live markets. For this study, DDQN's advantages of simplicity, interpretability, and reliable implementation outweigh TD3's theoretical sophistication.

3.2 Problem Formalisation

This section establishes the theoretical foundation by formulating futures roll trading as a Markov Decision Process (MDP), accounting for the agent receiving information, choosing actions, and receiving rewards in iterative cycles (Puterman, 1994).

3.2.1 Markov Decision Process

The trading problem is considered an MDP. The process starts with the agent receiving information about the environment denoted as state S_t. Based on that state, the agent chooses an action A_t and a reward R_t is given to the agent in the next time step. The agent then gets into another new state S_{t+1} and gets into another round of action, reward, and state. The loop of interaction between the agent and the environment produces trajectory

$$\tau = [S_0, A_0, R_0, S_1, A_1, R_1, S_2, A_2, R_2, \ldots]$$

The goal of RL is to maximize the expected cumulative value of reward at any time t, which is denoted as G_t with DDQN optimisation:

$$G_t = E\left[\sum_{k=t+1}^{T} \gamma^{k-t-1} R_k\right]$$

Where $\gamma = 0.9900$ (dimensionless discount factor), R_k is reward at step k (dollars), T is terminal time step (days), and $E[\cdot]$ denotes expectation.

3.2.2 State Space Design

The state space combines market features with trading state information to provide comprehensive environmental awareness for the agent. The state representation is defined as:

$$s_t = [s_t^{market}, s_t^{trading}] \in \mathbb{R}^{(25L+8)}$$

where $s_t^{market} \in \mathbb{R}^{25\times L}$ is flattened to $\mathbb{R}^{25L}$ and $s_t^{trading} \in \mathbb{R}^8$, representing L timesteps of 25 market features, where $s_t^{trading} \in \mathbb{R}^8$ represents the trading state. The lookback window length L varies by commodity to accommodate different rolling cycles (see Appendix B for specific values).

Market Features Component: The market features are structured as:

$$s_t^{market} = [x_{t-L+1}, x_{t-L+2}, \ldots, x_t]$$

where each $x_i \in \mathbb{R}^{25}$ contains normalized market variables:

$$x_i = [p_i^{open1}, p_i^{high1}, p_i^{low1}, p_i^{close1}, v_i^1, OI_i^1, p_i^{open2}, p_i^{high2}, p_i^{low2}, p_i^{close2}, v_i^2, OI_i^2, \ldots, \Delta OI_i^{ratio}]$$

These features include OHLC prices (dollars per contract), volume (v_i, number of contracts), open interest (OI_i, number of contracts), calendar spreads, and their respective ratios and changes for both front-month and second-month contracts.

Trading State Component:

The trading state captures current portfolio information:

$$s_t^{trading} = \left[\frac{pos_t}{pos_{max}}, \frac{PnL_t^{unrealized}}{K}, \frac{PnL_t^{total}}{K}, \frac{PnL_t^{daily}}{K}, \frac{equity_t}{K}, drawdown_t, stop_{triggered}, \frac{t}{T} \right]$$

where pos_t is the current position (number of contracts), $pos_{max} = 8$ contracts is the maximum allowed position, K is the initial capital ($) for normalization, PnL_t represents various profit and loss measures ($), $equity_t$ is current account equity ($), $drawdown_t$ is the current drawdown ($), $stop_{triggered} \in \{0, 1\}$ indicates stop-loss status, and $\frac{t}{T}$ represents progress through the dataset (dimensionless).

This state representation addresses the multi-scale nature of commodity trading decisions, where both historical pattern recognition and current portfolio state significantly influence optimal action selection. The market features capture the sequential dependencies essential for identifying roll patterns and market transitions, while the instantaneous trading features ensure decision-making accounts for existing exposure and performance constraints.

State Preprocessing

The state vector s_t undergoes preprocessing before network input to accommodate the different architectural requirements of the LSTM and fully connected components:

$$s_t = [s_t^{market}, s_t^{trading}] \in \mathbb{R}^{(25L+8)} \tag{3.1}$$

$$s_t^{market} : \mathbb{R}^{25L} \to \mathbb{R}^{L \times 25} \text{ (reshape)} \tag{3.2}$$

$$s_t^{market} : \mathbb{R}^{L \times 25} \to \mathbb{R}^{25 \times L} \text{ (transpose)} \tag{3.3}$$

$$s_t^{trading} : \mathbb{R}^8 \text{ (unchanged)} \tag{3.4}$$

where the reshaped market features maintain temporal relationships for LSTM processing while the trading state vector is directly fed to fully connected layers. Figure 3.1 illustrates the state transformation from the flat MDP representation to the neural network inputs.

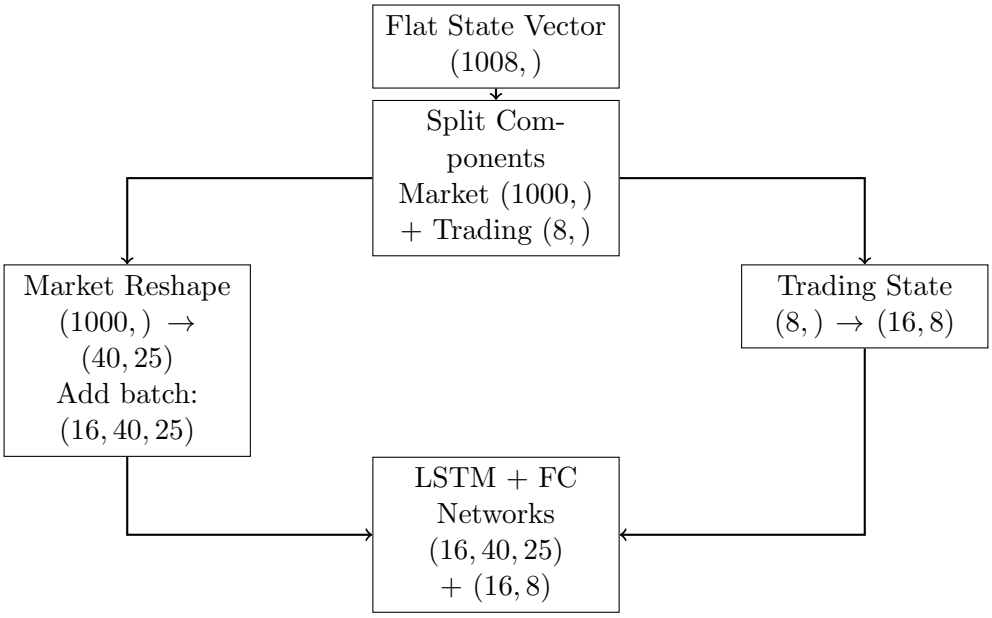

Figure 3.1: State preprocessing pipeline for neural network input

3.2.3 Action Space Definition

The action space is defined as a discrete set representing position changes as fractions of maximum position size:

$$\mathcal{A} = \{a_0, a_1, a_2, a_3, a_4\} = \{-0.5, -0.25, 0.0, 0.25, 0.5\}$$

where each action $a_t \in \mathcal{A}$ represents a position change:

$$\Delta pos_t = a_t \times pos_{max}$$

with $pos_{max} = 8$ contracts. The resulting new position is constrained by:

$$pos_{t+1} = \text{clip}(pos_t + \Delta pos_t, -pos_{max}, pos_{max})$$

This allows neutral positions as well as buying and selling in predefined increments of 50% of maximum position size. Actions are selected using the ϵ-greedy policy to balance exploration (selecting random actions with probability ϵ) versus exploitation (choosing actions with highest Q-values with probability $1-\epsilon$). The policy is formally defined as:

$$\pi(a|s) = \begin{cases} \frac{\epsilon}{|A|} + (1-\epsilon) & \text{if } a = \arg\max_{a'} Q(s, a'; \theta) \\ \frac{\epsilon}{|A|} & \text{otherwise} \end{cases}$$

where $|\mathcal{A}| = 5$ is the size of the action space. This strategy optimizes the exploration-exploitation trade-off during training. This strategy optimizes the exploration-exploitation trade-off during training (Dann et al., 2022).

The discrete structure enhances training stability by providing clear decision boundaries and reduces the complexity of the exploration-exploitation trade-off compared to continuous action spaces, making it suitable for the systematic nature of roll trading.

3.2.4 Reward Function Design

The reward function in this study is a weighted linear combination of performance metrics and behavioral constraints.

$$R(t) = \alpha_1 \cdot \frac{\text{P\&L}(t)}{K} + \alpha_2 \cdot \left(\frac{\text{WR}(t)}{100} - 0.5000 \right) \cdot 2.0000 - \alpha_3 \cdot \frac{\text{RP}(t)}{\text{RP}_{max}}$$

where K is initial capital (\$), $\text{WR}(t)$ is win rate (%), and RP_{max} is maximum acceptable risk penalty

and α_1, α_2, and α_3 are the weights for each metric (defined in Appendix B).

The primary component, α_1, uses daily profit-and-loss to provide consistent learning signals across different portfolio sizes.

The win rate component $\left(\frac{WR(t)}{100} - 0.5000\right) \cdot 2.0000$ centers around a 50% win rate, providing positive rewards for win rates above 50% and penalties below 50%. This encourages trading consistency by rewarding strategies that maintain success rates above 50%, addressing the common problem where algorithms may pursue high-risk, infrequent trades rather than developing sustainable trading patterns.

The risk penalty component dynamically penalises excessive drawdowns when losses exceed predefined thresholds, promoting risk-adjusted performance rather than pure return maximisation. This approach incorporates realistic criteria used in trading to assess the success of a trading strategy within the constraints of real market dynamics.

3.3 Data Construction and Features

3.3.1 Data Selection

The dataset comprises daily price data for the two nearest-month futures contracts within the GSCI, obtained from Bloomberg to ensure data quality and consistency. Daily frequency provides the optimal balance between meaningful roll timing signals and noise. See table of commodities to be examined below in Table 3.1.

Table 3.1: GSCI constituent contracts used in this study

Energy	**Softs**	**Metals**	**Livestock**
Brent Oil & WTI*	Sugar #11	Silver & Gold	Live Cattle

* West Texas Intermediate

3.3.2 Historical Coverage

The dataset provides almost 15 years of market data from January 2010 to May 2025 inclusive. This period captures multiple commodity cycles - illustrated below in Figure 3.2 - including the commodity downturn (2014-2016), the Covid pandemic (2020-2021) and the Russian invasion of Ukraine (2022) ensuring the algorithm trains across diverse market regimes.

Figure 3.2: Goldman Sachs Commodities Index 2010 - 2025 (TradingView, 2025).

3.3.3 Data Features

The following variables make up the dataset: Bloomberg daily open, high, low, close (OHLC) price data, volume and open interest. Features added include the calendar spread (front month - second month), volume and open interest changes, and ratios between front and back contracts. This feature selection reflects the critical importance of using economically meaningful signals essential to this strategy rather than arbitrary technical indicators. As stated in literature review, performance depends on using high-quality, meaningful signals. Adding just two noise variables can cut performance in half. Complexity without economic foundation leads to inferior results rather than improved outcomes (Portfolio Solutions Group, 2024). A comprehensive specification of all 25 input features is provided in Appendix A.5 (Table A.16).

3.4 Algorithm Implementation

This section details the mathematical formulations, architectural specifications, and implementation details for the LSTM network and the DDQN algorithm for decision making.

3.4.1 LSTM Network Design

LSTM architecture employs a gating mechanism that regulates information flow through time, enabling selective retention and forgetting of temporal patterns (shown in Figure 3.3). This capability is particularly relevant for roll trading, where trading signals develop over extended periods (days and weeks) and exhibit complex temporal interdependencies The LSTM cell state evolution is governed by three multiplicative gates that control information flow:

Forget Gate:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

Input Gate:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Output Gate:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

Cell State Update:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$h_t = o_t * \tanh(C_t)$$

Where σ represents the sigmoid activation function, W denotes weight matrices, b represents bias vectors, x_t is the input at time t (dimensionless), h_t is the hidden state at time t (dimensionless), C_t is the cell state at time t (dimensionless), brackets $[\cdot, \cdot]$ denote vector concatenation and $*$ indicates element-wise multiplication. The complete LSTM architecture, including input variables, learnable parameters, and logic flow, is detailed in Appendix A.2 (Tables A.2–A.6).

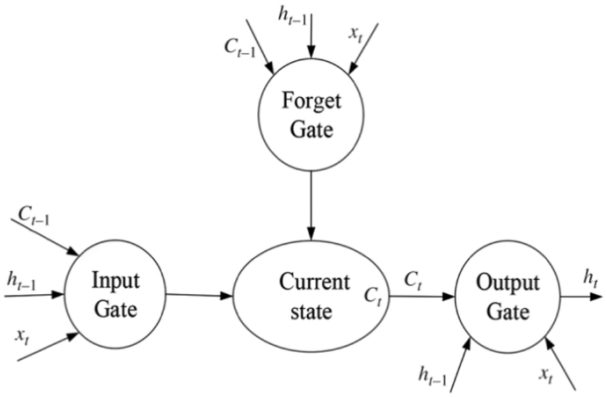

Figure 3.3: Structure of an LSTM unit (Ding et al., 2015).

3.4.2 Architecture Specifications

The LSTM network employs a hierarchical configuration. The architecture consists of 2 stacked LSTM layers, each containing 32 neurons. The network processes input sequences of shape (16, 30, 25), where 16 represents batch size, 40 represents the lookback window (variable as each commodity has a different rolling cycle) and 25 encompasses the market feature dimensions. Post-LSTM processing utilizes two fully connected layers with dimensions 32 and 16 respectively, incorporating ReLU activation and 0.05 dropout to refine temporal features for trading decision integration. This configuration provides approximately 30,000 trainable parameters, appropriate for the available data volume while maintaining sufficient model complexity for roll pattern recognition.

3.4.3 Implementation Considerations

Weights are initialised using Xavier uniform distribution to stabilise gradient flow during early training phases (Glorot and Bengio, 2010). Cell state vectors are initialised as zero tensors at each sequence start, ensuring consistent processing across training batches. Gradient clipping with threshold = 1.0000 prevents exploding gradients by normalizing the gradient vector when its norm exceeds the threshold:

$$\nabla\theta \leftarrow \frac{\nabla\theta}{\max(1, \|\nabla\theta\|_2)}$$

where $\|\nabla\theta\|_2 = \sqrt{\sum_i (\nabla\theta_i)^2}$ is the L2 norm of the gradient vector.

This limits the magnitude of gradients during backpropagation to prevent them from becoming too large and destabilising the training process, maintaining stable learning particularly during extreme market volatility such as the 2020-2022 energy crisis.

A dropout rate of 0.0500 is applied. Dropout randomly sets a percentage of neurons to zero during training to prevent the network from becoming overly dependent on specific neurons, forcing it to learn more robust patterns (Zaremba et al., 2014). Bidirectional configuration is disabled to maintain temporal causality constraints essential for trading applications, ensuring the network cannot access future information during pattern recognition.

3.4.4 DDQN Design

As per the literature review, DDQN addresses the overestimation bias inherent in standard DQN by employing two distinct networks: an online network θ for action selection and a target network θ^- for action evaluation.

The algorithm learns by minimizing the Mean Squared Error between predicted Q-values and target values:

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

The key innovation of DDQN is the decoupling of action selection from action evaluation. During training, the online network selects the optimal action for the next state:

$$a^* = \arg\max_{a'} Q(s', a'; \theta)$$

while the target network evaluates this selected action to compute the target value:

$$y = r + \gamma(1 - \text{done}) \cdot Q(s', a^*; \theta^-)$$

where done $= \begin{cases} 1 & \text{if episode terminates} \\ 0 & \text{otherwise} \end{cases}$

This formulation ensures consistency with the MDP framework established in Section 3.2.1, where the discount factor $\gamma = 0.9900$ applies only to non-terminal states. The binary indicator $(1 - \text{done})$ implements the terminal condition, setting future rewards to zero when done $= 1$ (episode termination) and maintaining the discounted future value when done $= 0$ (episode continuation).

This mechanism effectively separates action selection from value estimation, significantly reducing overestimation bias in volatile trading environments where inflated Q-values could lead to overly aggressive position-taking, detrimental to risk management.

3.4.5 Target network update strategy

The target network update mechanism balances training stability with policy adaptation through a periodic hard update strategy. Unlike soft updates, which continuously blend online and target network parameters, the hard update approach completely copies online network weights to the target network every 200 training steps. This prevents the moving target problem that destabilizes Q-learning convergence while allowing sufficient time for the online network to learn from stable target values. During the 200-step intervals between updates, the target network provides consistent evaluation targets:

$$y_t = r_t + \gamma Q(s_{t+1}, a^*; \theta^-)$$

enabling stable gradient computation for the online network optimization. The target network parameters remain frozen between updates to prevent inadvertent parameter modifications during backpropagation. This approach provides a balance between stability and adaptation speed appropriate for situations where market regime changes require policy updates but excessive target network volatility could destabilise learning fundamental roll patterns.

3.5 Training: Learning Through Experience

This section outlines the methodology in training the algorithm and assessing its performance against the classic strategy benchmark.

3.5.1 Training Environment Construction

The trading environment transforms the historical data into a structured learning framework where the agent can explore trading strategies without capital risk. The environment simulates realistic market conditions by incorporating:

- **Transaction costs**: variable to reflect different fees per product
- **Slippage**: variable per contract to reflect real-world trading
- **Position limits**: 8 contracts max position size for simplicity
- **Risk controls**: Daily loss limits and max drawdown limit

State representation combines temporal market features with instantaneous portfolio information, providing comprehensive environmental awareness for decision-making.

3.5.2 Training Process

At each training step, the following process is executed:

1. **Action Selection**: The online network selects actions using the ϵ-greedy policy
2. **Environment Interaction**: Execute the chosen action and observe the resulting reward and next state
3. **Experience Storage**: Store the complete transition tuple in the experience replay buffer
4. **Batch Sampling**: Randomly sample a batch of experience tuples from the replay buffer
5. **Target Computation**: Apply the DDQN mechanism
6. **Network Optimization**: Update the online network weights
7. **Target Network Updates**: Hard update target network every 200 training steps

This process repeats for the preset number of training episodes. After training, the online network generates trading decisions by selecting actions with the highest Q-values, ensuring that decisions incorporate both market patterns learned by the LSTM and immediate portfolio constraints for context-aware position management.

3.5.3 Training Strategy

Data Split

The dataset employs a chronological 70/10/20 split for training, validation, and testing respectively to maintain temporal integrity essential for financial time series analysis. This ensures no look-ahead bias, where the model trains exclusively on historical data preceding the validation and test periods, mimicking real-world trading conditions where future information is unavailable (López de Prado, 2018). The 70% training allocation provides sufficient historical data for the algorithm to learn market patterns, while the 10% validation set enables hyperparameter optimisation without overfitting to the test data. The final 20% test allocation offers an unbiased evaluation of strategy performance on genuinely out-of-sample data, providing realistic estimates of live trading performance.

Training Runs

Henderson et al.'s (2018) work delivers a crucial reality check for deep RL: many reported algorithmic improvements may be artifacts of experimental choices rather than genuine advances. Their analysis reveals that hyperparameter selection, random seed variance, environment choice, and implementation details can overwhelm claimed algorithmic improvements based solely on network architecture choices, showing statistically significant differences between identical algorithms using different random seeds. To counter this they propose to train an algorithm 5 times from scratch using different random seeds for each run. After training, backtest each of the 5 trained models and measure the average performance and variability across all 5 models This approach ensures results are statistically meaningful rather than just one potentially lucky or unlucky training outcome.

In total, this will mean 30 training cycles to ensure statistical robustness.(6 commodities, 5 independent training runs each). Preliminary test runs on partial datasets require approximately 15 minutes each on available hardware. To expedite this process, cloud GPU services will be utilised, specifically Paperspace's H100 instances which dramatically reduce training time and enable completion within a practical timeframe (Paperspace, 2025).

3.6 Validation

The validation process serves to guide model selection, tune hyperparameters, and assess generalisation prior to out-of-sample testing. By evaluating performance on a temporally separated validation set, validation ensures that observed improvements are not the result of overfitting or favourable market conditions during training.

3.6.1 Tuning

Ablation Studies

Ablation studies test individual components such as LSTM and DDQN hyperparameters and different reward function formulations on validation data to isolate which specific design choices actually drive trading performance improvements.

Preliminary ablation studies highlighted two critical dependencies. First, reward function weighting exerted non-linear effects: configurations prioritising P&L ($\alpha_1 > 0.7000$) incentivised excessive trading, amplifying cost erosion despite nominally higher gross returns. Second, the lookback window (L) exhibited a Goldilocks effect: shorter horizons ($L < 15$) failed to capture full roll cycles, while extended windows ($L > 80$) introduced noise from obsolete market regimes. Each commodity employs a different lookback window length due to varying rolling cycles across markets. See Appendix A.1 for the specific lookback window used for each commodity.

These observations mirror Kabbani and Duman's 2022 findings on hyperparameter fragility in RL. Even theoretically sound architectures require calibration to navigate noisy market environments. Given these sensitivities, hyperparameters are systematically evaluated across the following ranges before finalising the architecture.

Hyperparameter Tuning

The tuning strategy combined coarse grid search with manual refinement informed by domain-specific priors and prior literature Kabbani and Duman (2022).

- **Learning rate (α)**: Tested values $\{0.0005, 0.0001, 0.00005\}$; fixed at $\alpha = 0.0001$ to ensure gradient stability over longer training horizons and reduce volatility in Q-value updates. Lower rates avoided divergence, while higher rates caused unstable updates.
- **LSTM architecture**: Evaluated {2 layers, 32 units}, {2 layers, 64 units} and {3 layers, 128 units}. The final configuration used 2 layers with 32 hidden units, providing sufficient capacity without overfitting, particularly in thinly traded contracts.
- **Dropout rate**: Explored $\{0\%, 5\%, 10\%\}$; applied at 5% to discourage feature co-adaptation while preserving temporal sensitivity. Higher rates degraded performance, while 0% led to mild overfitting.
- **ϵ-greedy policy**: Tested decay rates $\{0.9900, 0.9950, 0.9990\}$ with $\epsilon_{\text{start}} = 1.0000$ and $\epsilon_{\text{end}} = 0.2000$. Selected $\epsilon_{\text{decay}} = 0.9950$ for slow but steady exploration-to-exploitation transition. Faster decays (0.9990) caused premature convergence to suboptimal policies.
- **Target network update frequency**: Compared $\{100, 200, 500\}$ steps; chose 200 steps as a compromise between policy stability (fewer updates) and adaptability (frequent updates). Shorter intervals increased variance, while longer intervals delayed convergence.
- **Mini-batch size**: Tried $\{8, 16, 32\}$; fixed at 16, consistent with prior DRL financial studies. Smaller batches (8) increased noise, while larger batches (32) strained memory resources without improving stability.

Taken together, the tuning strategy aimed to strike a balance between model flexibility and regularisation, informed by both empirical performance and the economic structure of the problem domain. A full listing of hyperparameter values is provided in Appendix A.3.

3.6.2 Benchmark Strategy

The benchmark for this study will be the strategy examined by Mou (2011) and Irwin et al. (2022): front-running the 'Goldman Roll' by shorting the expiring (front-month) contract and going long the next (deferred) contract just before the roll window. Mou found that this achieved a Sharpe ratio of 4.39 and returns of 3.6% during the early 2000s. Irwin et al. confirmed the profitability of this strategy, but noted that such opportunities largely disappeared after 2012.

4. Testing & Results

4.1 Backtesting Environment

The backtesting environment simulates realistic trading conditions, addressing the critical limitations identified in existing literature. This incorporates all major cost components and risk constraints encountered in live commodity futures trading.

Market Friction Implementation

The environment models realistic transaction and slippage costs observed through direct trading experience across commodities markets. Transaction costs at $0.75 per contract, a commission structures observed at trading firms. Bid-ask slippage is implemented at 0.5 ticks per contract to simulate market impact. This captures execution costs typically resulting in approximately one tick loss between trade entry and exit.

Risk Management Framework

Designed to prevent capital destruction, daily loss limits are set at $500, triggering liquidation when exceeded to simulate stop-loss procedures. A maximum drawdown threshold of 30% triggers immediate position closure and strategy termination, reflecting institutional risk management practices. A position hold timer mechanism prevents indefinite position holding by forcing position closure after 10 trading days, addresses the tendency of RL agents to hold losing positions indefinitely.

Performance Measurement

The environment maintains trade statistics including win rate calculations, profit factor metrics, and detailed equity curve tracking. Each completed trade is classified as winning, losing, or break-even. Running statistics enable real-time monitoring of strategy performance during backtesting.

Goldman Roll Timing Constraints

The environment incorporates specific business day calculations essential for Goldman Roll strategy implementation. Business days are computed accurately excluding weekends, with roll windows defined from the 5th to 9th business day of each month reflecting actual GSCI rebalancing schedules.

Statistical Validation Framework

Following the Henderson et al. (2018) methodology for statistical significance in deep reinforcement learning validated in Section 3.5.3, the backtesting framework executes multiple independent training runs with different random seeds. This approach ensures that reported performance metrics represent statistically meaningful results rather than artifacts of favorable initialisation conditions.

4.2 Performance Metrics

Table 4.1 presents the backtesting results for the LSTM-DDQN trading system across the commodity futures contracts, with each result representing the mean and standard deviation across five independent training runs using 5 different random seeds. Detailed per-commodity results, including individual training runs and benchmark comparisons, are tabulated in Appendix A.4 (Tables A.9–A.15).

Table 4.1: DDQN Trading System Performance Metrics

Commodity	**Total Return (%)**	**Win Rate (%)**
Crude Oil (CL)	-29.72 ± 0.33	42.4 ± 7.7
Brent Oil (CO)	-29.88 ± 0.30	46.8 ± 7.7
Gold (GC)	-8.67 ± 3.20	60.2 ± 8.5
Live Cattle (LC)	-5.75 ± 10.37	58.9 ± 14.3
Silver (SI)	-30.19 ± 0.15	50.4 ± 7.2
Sugar #11 (SB)	-30.26 ± 0.13	44.8 ± 9.7
Average	$\mathbf{-22.41 \pm 11.81}$	$\mathbf{50.6 \pm 7.4}$

Performance measured over out-of-sample test period (2022-2025).

Statistical significance testing following Henderson et al. (2018) reveals that performance differences between commodities are statistically significant at the 95% confidence level (detailed confidence intervals in Appendix A.4 (Table A.8)).

4.3 Key Findings

The results presented in Table 4.1 demonstrate uniformly negative returns across all tested commodities, with an average loss of 30% reaching the maximum drawdown limit.

Transaction Cost Erosion

While several runs demonstrated nominally profitable performance prior to execution costs - with instances achieving positive raw P&L and win rates exceeding 50% (notably, select Gold runs: 63.1% win rate; Crude Oil: 57.6%) - these superficially favorable metrics are entirely negated when accounting for transaction costs and slippage.

Individual trade records consistently show positive gross P&L subsequently eroded by transaction costs. This pattern was particularly pronounced in Brent Oil Run 2, which achieved a gross profit factor of 1.38 before costs, yet delivered substantial net losses.

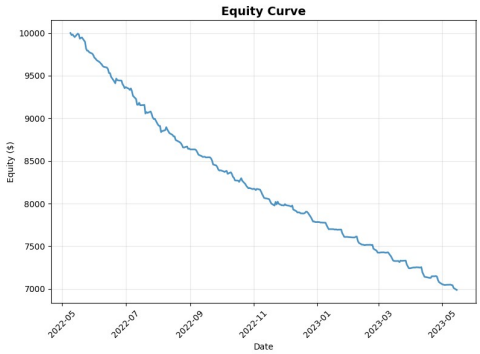

Figure 4.1: Equity curve for Brent Oil Training Run #2.

Although the equity curve for Brent Oil Run 2, presented in Figure 4.1, shows intermittent small spikes — indicating isolated profitable trades net of costs — the overall trajectory is a consistent decline. The frequent position adjustments inherent in the LSTM-DDQN strategy amplify costs, turning what might have been a marginally profitable strategy into one with significant net losses. Even commodities like Gold, which exhibited a win rate of 60.2% presented in Table 4.1, failed to overcome the cumulative drag of market frictions.

While certain runs exhibited favorable risk-reward characteristics pre-cost, the many position adjustments and lack of asymmetric payoff structures led to negative net retruns. This phenomenon exemplifies transaction cost fragility, wherein statistically advantageous strategies fail under realistic market friction.

4.3.1 Benchmark Strategy Comparison

Table 4.2 presents a comparative analysis between the LSTM-DDQN trading system and traditional Goldman Roll strategies implemented following Mou (2011), evaluated over the same test period (2022-2025).

Table 4.2: Benchmark Strategy Performance Comparison

	Goldman Roll Strategy		**LSTM-DDQN Strategy**	
Commodity	**Return (%)**	**Win Rate (%)**	**Return (%)**	**Win Rate (%)**
Crude Oil (CL)	−4.47	0.0	−29.72	42.4
Brent Oil (CO)	−4.61	0.0	−29.88	46.8
Gold (GC)	−1.02	45.9	−8.67	60.2
Live Cattle (LC)	−4.39	45.9	−5.75	58.9
Silver (SI)	−4.23	0.0	−30.19	50.4
Sugar (SB)*	−4.19	0.0	−30.26	44.8
Average	**−3.81**	**15.3**	**−23.09**	**49.7**

Relative Performance Analysis

Despite the LSTM-DDQN system's overall negative returns, its ability to identify profitable trades represents a critical finding.

The benchmark strategy - the same approach that once generated Sharpe ratios of 4.39 Mou (2011) - failed to achieve a single winning trade in energy and metals markets (0% win rate for Crude Oil, Brent Oil, and Silver) for the test period. In stark contrast, the LSTM-DDQN system achieved win rates of 42.4–60.2% as shown in Table 4.2, demonstrating its capacity to detect residual roll patterns where the benchmark strategy collapsed entirely.

This divergence is especially notable in Gold and Live Cattle markets, where the LSTM-DDQN's win rates (60.2% and 58.9%, respectively) surpassed the benchmark's 45.9%. While transaction costs eroded net profitability, the algorithm's consistent ability to identify winning trades suggests that modern roll patterns may require adaptive, data-driven methods to exploit fleeting opportunities.

The benchmark's total failure in energy markets (0% win rate) versus the LSTM-DDQN's near-50% win rate underscores a fundamental shift: historical roll signals are extinct, but machine learning can still uncover latent inefficiencies. Future work could refine the LSTM-DDQN's cost sensitivity or combine its pattern recognition with lower-frequency execution to preserve alpha.

Market Efficiency and Strategy Obsolescence

The Goldman Roll benchmark results offer compelling evidence for the strategy's obsolescence in modern markets. All six commodities recorded negative returns over the test period, averaging a 3.26% loss. Particularly striking is the complete failure in energy and metals markets (Oil and Silver) where zero winning trades were observed across 37 transactions each, translating to 100% loss rates

This uniform failure across diverse commodity sectors supports the literature's assertion that Goldman Roll arbitrage opportunities have been systematically eliminated since 2012. The strategy that once delivered Sharpe ratios of 4.39 during 2000-2010 Mou (2011) now produces consistently negative returns. The disappearance of these patterns is widely attributed to increased algorithmic participation and capital allocation to systematic trading, leading to the erosion of predictable price distortions once created by GSCI rebalancing flows.

The fact that both the LSTM-DDQN system and benchmark strategies failed across most commodities further underscores a broader market transformation. The results indicate that historical roll flows—previously exploitable by discretionary traders—have likely become fully priced in by modern, high-frequency participants.

However, not all markets behaved uniformly. Gold and Live Cattle stood out with the smallest performance gaps between the LSTM-DDQN agent and the benchmark, hinting at potential residual inefficiencies or slower algorithmic adoption curves. These commodities are known for idiosyncratic structural features (such as geopolitical sensitivity in gold or lower liquidity in cattle) which may preserve subtle, harder-to-exploit roll patterns. This variation suggests that some markets may still exhibit exploitable structure, albeit faint and likely dependent on fragile hyperparameter configurations.

In contrast, the total collapse in energy markets (e.g., WTI and Brent) mirrors Irwin et al. (2022), who document an 80% collapse in roll-related alpha post-2012, attributing it to aggressive algorithmic penetration and competitive cost arbitrage.

Furthermore, non-stationarity in roll dynamics, particularly after structural shifts in index fund behavior and regulatory reforms, presents additional challenges. These shifts limit the ability of data-driven agents to generalize across regimes, an issue echoed in the findings of Kabbani and Duman (2022) on the fragility of financial RL models in non-stationary environments.

The broader implications of these findings, including considerations for alternative modeling frameworks and the evolving nature of market efficiency, are discussed in Section 6.

5. Legal, Social, Ethical and Professional Issues

This chapter examines the ethical considerations arising from developing deep reinforcement learning algorithms for commodity futures trading, checked against professional computing standards as required for responsible AI development.

BCS Code of Conduct Clause-by-Clause Analysis

Following the BCS Code of Conduct for BCS Members (British Computer Society (BCS), 2022), each clause is examined for relevance to this algorithmic trading research:

Regulatory and Societal Impact Analysis

Financial Regulation: Under MiFID II Article 17, algorithmic trading systems require effective risk controls. This project addresses these requirements through embedded risk management and transparent validation methodologies that would facilitate regulatory oversight.

Market Impact: Commodity futures affect consumer prices for essential goods including food and energy. The finding that transaction costs eliminate algorithmic profitability suggests current market structure provides natural protection against excessive speculation. The complete failure of traditional Goldman Roll strategies (0% win rates in energy markets) demonstrates that historical arbitrage opportunities have been systematically eliminated, reducing artificial price pressures.

Systemic Risk: This project addresses concerns about algorithmic convergence through statistical validation showing high performance variability across training runs, indicating strategies are unlikely to achieve harmful synchronised behavior at scale.

Environmental and Technological Considerations

This project's computational requirements (30 independent training runs using cloud GPU infrastructure) highlight broader sustainability concerns about AI research energy consumption. However, the negative findings - demonstrating that sophisticated AI cannot overcome basic market frictions - provide valuable knowledge preventing wasted future research efforts in this domain.

Professional Accountability and Transparency

Following IET Rules of Conduct (Institution of Engineering and Technology (IET), 2019), this project maintains professional integrity through open-source implementation, comprehensive documentation, and transparent reporting of both capabilities and limitations. This transparency enables academic scrutiny while contributing to responsible AI development standards in financial applications.

Conclusion: This systematic analysis demonstrates compliance with professional computing standards while addressing broader ethical implications of algorithmic trading research. The Henderson et al. (Henderson et al., 2018) validation framework serves dual purposes: ensuring statistical rigor while preventing misleading claims about AI trading capabilities, ultimately serving public interest over private profit maximisation.

BCS Clause	Relevant	Application to Project
1.a Act in public interest	Yes	Incorporating realistic transaction costs prevents misleading claims that could harm retail investors. Using Henderson et al.'s 2018 statistical validation method using five independent runs ensures meaningful results rather than initialisation artifacts.
1.b Consider disabled persons	No	Not applicable - algorithmic trading systems do not directly interact with accessibility requirements.
1.c Protect environment	Yes	Project required 30 GPU training runs raising sustainability concerns about computational resource consumption relative to academic benefit.
1.d Respect cultural diversity	No	Not applicable - mathematical algorithms operate independently of cultural considerations.
2.a Demonstrate competence	Yes	Rigorous methodology combining DDQN + LSTM with comprehensive backtesting demonstrates technical competence in RL implementation.
2.b Keep skills current	Yes	Project incorporates latest deep learning advances (2018-2025 literature) and current market structure understanding.
2.c Not claim competence beyond skills	Yes	Transparent reporting of strategy failures and limitations prevents overstatement of algorithmic capabilities.
2.d Accept responsibility	Yes	Open-source implementation enables verification and accountability for algorithmic decisions and risk controls.
3.a Accept professional responsibility	Yes	Project includes embedded risk management (position limits, stop-losses) addressing systemic risk concerns.
3.b Exercise professional judgment	Yes	Balanced assessment showing where AI succeeds (pattern recognition) yet fails economically (transaction cost erosion).
3.c Avoid conflicts of interest	Yes	Academic project maintains independence from commercial trading interests.
3.d Accept duty to profession	Yes	Contributes to professional knowledge through methodological advancement and honest capability assessment.
4.a Have regard for health & safety	Yes	Risk controls prevent strategies contributing to market instability or systemic financial risk.
4.b Have regard for rights of others	Yes	Respects Bloomberg data licensing, uses open-source frameworks appropriately, provides full code attribution.
4.c Conduct activities with integrity	Yes	Transparent methodology, comprehensive validation, and honest reporting of negative results demonstrate research integrity.
4.d Respect confidentiality	No	Not applicable - project uses publicly available market data without confidential information.

Table 5.1: BCS Code of Conduct Relevance Analysis

6. Conclusions and Recommendations

This section summarises the outcomes of this project and their practical implications. The findings are presented as actionable recommendations for both trading system development and future technical work.

6.1 Key Findings

This work examines the application of AI methods - specifically Deep Reinforcement Learning (DRL) - to commodity futures roll trading, yielding several important technical findings.

- Pattern Recognition Capability: The LSTM-DDQN framework demonstrated statistically significant pattern recognition in commodity roll periods, achieving an average win rate of 50.6% across tested markets and surpassing the 0% win rate of traditional Goldman Roll strategies in energy markets. Notably, Gold (60.2%) and Live Cattle (58.9%) exhibited the strongest performance, suggesting residual inefficiencies persist in certain commodities.
- Transaction Cost Fragility: Despite its' predictive edge, the LSTM-DDQN's profitability was entirely eroded by realistic transaction costs and slippage. This highlights a critical limitation in deploying DRL for high-frequency trading strategies, where frequent position adjustments amplify costs.
- Obsolescence of Traditional Strategies: The complete collapse of the Goldman Roll benchmark (0% win rates in energy and metals markets) confirms that historical arbitrage opportunities have been systematically eliminated since 2012, as markets adapted to algorithmic competition.

These findings underscore the technical promise of DRL in identifying latent market patterns but also reveal significant practical barriers to deployment. The following subsections explore these limitations and propose actionable refinements for future research.

6.2 Limitations and Implementation Challenges

Several constraints in this work warrant careful consideration when interpreting the results. First, the daily frequency of the price data represents a significant limitation, as it cannot capture intraday patterns around settlement periods when institutional roll flows often manifest. As noted in practitioner interviews and direct experience, volume surges during specific intraday windows frequently signal roll activity, but such granular data was unavailable for this analysis. This temporal resolution constraint likely obscured important microstructure signals that could have improved strategy performance. Future work could incorporate intraday data to capture settlement-period roll flows.

Market structures have undergone substantial evolution since the benchmark period studied by Mou (2011). Notably, daily turnover in over-the-counter (OTC) derivatives more than doubled since 2016, capturing nearly half of total trading activity Ehlers and Hardy (2019). This has possibly diverted significant roll volume away from exchange-traded contracts, rendering exchange-based data less representative of true roll dynamics. This structural shift may partially explain the declining efficacy of traditional roll strategies observed in the results.

Additional limitations include:

- **Feature engineering constraints**: While the selected features captured standard roll indicators (open interest, calendar spreads), they may not fully represent modern market dynamics where roll patterns have become more nuanced.
- **Algorithm selection**: DDQN's discrete action space may be suboptimal compared to continuous alternatives like TD3 for position sizing.
- **Non-stationarity**: The 15-year training period encompasses multiple regime shifts (Section 3.3.2) and the algorithm's inability to dynamically adapt may have diluted recent patterns.
- **Benchmark comparability**: The traditional Goldman Roll strategy (Section 4.3.1) may no longer represent a viable baseline in today's markets, raising questions about appropriate performance benchmarks.

These limitations collectively suggest that while the methodological framework is sound, both data constraints and market evolution have likely impacted the strategy's potential effectiveness. They highlight the importance of continuous model adaptation in response to changing market microstructures.

6.3 Implications and Future Research Directions

The empirical results demonstrate that while the LSTM-DDQN framework achieved statistically significant win rate improvements over traditional strategies (particularly in Gold and Live Cattle markets, with 60.2% and 58.9% success rates respectively), the net profitability was eroded by transaction costs. This suggests that deep reinforcement learning retains value for *pattern recognition* in commodity roll periods, but requires structural modifications for practical implementation.

6.3.1 Pattern Recognition and market-Specific Performance

The LSTM-DDQN's ability to achieve 50.6% average win rates while Mou's (2011) benchmark strategy achieved 0% in most tested markets demonstrates pattern detection, stemming from three architectural advantages:

- The LSTM component's ability to detect early roll signals beyond the traditional 5th-9th business day window.
- The algorithm simultaneously processes multiple market variables identifying compound signals invisible to single-metric approaches.
- Most critically, continuous adaptation to evolving dynamics, unlike the static benchmark.

Market-specific results further highlight structural disparities. In energy markets (e.g., Crude Oil), high-frequency trading dominance and diverted institutional flows (Ehlers and Hardy, 2019)erode traditional roll signals. Conversely, agricultural markets (e.g., Live Cattle, 58.9% LSTM-DDQN win rate vs. 45.9% benchmark) retain inefficiencies due to lower liquidity, seasonal noise, and reduced systematic participation. This suggests algorithmic performance is contingent on market microstructure, with commodity-specific calibration offering potential gains.

6.3.2 Methodological Contributions

Beyond commodity-specific findings, this work establishes three methodological contributions to AI methods in financial markets:

- The dual-network preprocessing pipeline in Figure 3.3 that separates temporal market features for LSTM processing from instantaneous trading state for fully-connected layers represents a novel architecture for multi-scale financial decision-making.
- The comprehensive market friction modeling framework (Section 4.1) addresses a critical gap in financial RL literature by demonstrating how realistic transaction costs can completely negate theoretically profitable strategies. While most academic studies report inflated performance by omitting these costs, this work establishes a rigorous backtesting standard that includes bid-ask slippage, commission structures, and position-holding constraints.
- The economic feature selection methodology demonstrates that domain expertise significantly outperforms arbitrary indicators, confirming AQR's Portfolio Solutions Group's (2024) findings while providing a concrete framework for feature engineering in commodity markets.

These contributions extend beyond commodity trading to any sequential decision-making problem where temporal patterns interact with instantaneous constraints.

6.3.3 Future Research Opportunities

The transaction cost fragility and pattern recognition capabilities identified in this work point to four immediate research directions that could restore profitability to RL-based roll strategies:

- **Cost-Adaptive Reward Engineering**: Dynamically scaled penalties based on execution slippage could align incentives with net profitability rather than gross returns, addressing the core limitation identified in Section 4.3.
- **High-Frequency Data Integration**: Incorporating intraday data to capture settlement-period institutional flows may preserve exploitable alpha obscured by daily frequency limitations.
- **Continuous Action Spaces**: Implementation of Twin Delayed DDPG (TD3) could enable optimal position sizing while maintaining double Q-learning benefits.
- **Alternative Market Applications**: Extension to non-GSCI indices (Bloomberg Commodity Index, equity index futures) could validate broader applicability across different microstructure characteristics.

These refinements could bridge the gap between theoretical pattern recognition advantages and real-world deployability, ultimately unlocking the untapped potential of RL in futures roll trading.

Concluding Remarks

This work demonstrates that while traditional commodity roll arbitrage has been refined away by algorithmic competition, sophisticated machine learning approaches can still extract signals from these systematic flows. The LSTM-DDQN's ability to achieve superior win rates compared to completely failed benchmark strategies reveals that exploitable patterns persist, albeit in increasingly subtle forms that require adaptive, data-driven detection methods.

However, the crude reality of transaction costs serves as a sobering reminder that theoretical profitability means little without accounting for market frictions. The gap between gross pattern recognition success and net trading failure illustrates a fundamental challenge in algorithmic trading: identifying alpha is only half the battle, preserving it through execution is where many strategies strike out.

The market's evolution from easily exploitable Goldman Roll opportunities to today's algorithmically contested environment reflects broader trends in market efficiency. Yet this project's findings suggest that rather than complete market saturation, we observe a migration of opportunities toward more sophisticated analytical approaches. While human discretionary traders have been priced out of these markets, algorithms deploying AI methods demonstrate the potential to mine residual inefficiencies that simpler systematic approaches cannot detect.

Future development would be well-advised to focus not just on pattern recognition capabilities, but on the integration of execution costs into reward formulations to ensure algorithms don't just strike oil in backtests, but can actually refine profitable strategies for live deployment. The fusion of advanced pattern recognition with practical trading constraints may yet yield golden opportunities in commodity markets, provided the lessons of cost, frequency, and market microstructure evolution are properly integrated into algorithmic design.

Bibliography

Bagheri, M. (2024). Optimizing quantitative trading: An experimental study of dqn trading strategies and utility functions. Master's thesis, Tilburg University, Tilburg.

Brim, A. and Flann, N. S. (2022). Deep reinforcement learning stock market trading, utilizing a cnn with candlestick images. *PLOS ONE*, 17(2):e0263181. Accessed: 10 June 2025.

British Computer Society (BCS) (2022). Code of conduct for bcs members. Version 8, last reviewed and approved by Trustee Board 8 June 2022.

Chavan, S., Kumar, P., and Gianelle, T. (2021). Intelligent investment portfolio management using time-series analytics and deep reinforcement learning. *SMU Data Science Review*, 5(2):1–29. Accessed: 10 June 2025.

Dann, C., Mansour, Y., Mohri, M., Sekhari, A., and Sridharan, K. (2022). Guarantees for epsilon-greedy reinforcement learning with function approximation. In *Proceedings of Machine Learning Research*, volume 162, pages 4666–4689. Presented at the 39th International Conference on Machine Learning (ICML 2022), Baltimore, United States, 17–23 July.

Ding, X., Zhang, Y., Liu, T., and Duan, J. (2015). Deep learning for event-driven stock prediction. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)*, pages 2327–2333. Accessed on 30th Oct, 2023.

Ehlers, T. and Hardy, B. (2019). The evolution of OTC interest rate derivatives markets. *BIS Quarterly Review*, 2019:69–82. BIS Quarterly Review, December 2019, pp. 69–82.

Gao, L. (2024). Comparison of dqn and double dqn reinforcement learning algorithms for stock market prediction. In Ahmad, B. H., editor, *Proceedings of the 2023 International Conference on Data Science, Advanced Algorithm and Intelligent Computing (DAI 2023)*, volume 180 of *Advances in Intelligent Systems Research*, pages 168–177, Paris. Atlantis Press.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS 2010)*, pages 249–256.

Gu, S., Kelly, B., and Xiu, D. (2020). Empirical asset pricing via machine learning. *The Review of Financial Studies*, 33(5):2223–2273.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. (2018). Deep reinforcement learning that matters. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1):3207–3214. Accessed: 4 June 2025.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Institution of Engineering and Technology (IET) (2019). Rules of conduct. Approved by the Board of Trustees on 3 October 2019.

Irwin, S. H., Sanders, D. R., and Yan, L. (2022). The order flow cost of index rolling in commodity futures markets. *Applied Economic Perspectives and Policy*.

Israel, R., Kelly, B. T., and Moskowitz, T. J. (2020). Can machines "learn" finance? *Journal of Investment Management*, 18(2):23–41.

Kabbani, T. and Duman, E. (2022). Deep reinforcement learning approach for trading automation in the stock market. Technical report, Department of Industrial Engineering, Özyeğin University, Istanbul.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25:1097–1105.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lee, J. (2025). Aqr bets on machine learning as cliff asness becomes ai believer.

López de Prado, M. L. (2018). *Advances in Financial Machine Learning*. John Wiley & Sons, Hoboken, NJ.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Moghar, A. and Hamiche, M. (2020). Stock market prediction using lstm recurrent neural network. *Procedia Computer Science*, 170:1168–1173.

Mou, Y. (2011). *Limits to Arbitrage and Commodity Index Investment: Front-Running the Goldman Roll*. PhD thesis, Columbia University, New York.

Mussema, Y. and Kakeba, K. (2025). Developing an intelligent trading model for the ethiopia commodity exchange (ecx) using deep reinforcement learning algorithms. *Irish Interdisciplinary Journal of Science & Research (IIJSR)*, 9(1):23–35.

Nivedha, S., Rajaguhan, A., and Prabin, R. (2024). Atari pong with dqn and ddqn. *Journal of Emerging Technologies and Innovative Research*, 11(5):f295–f305. Accessed: 8 June 2025.

Paperspace (2025). Paperspace. Accessed: 24 June 2025.

Phuoc, T., Anh, P. T. K., Tam, P. H., and Nguyen, C. V. (2024). Applying machine learning algorithms to predict the stock price trend in the stock market – the case of vietnam. *Humanities and Social Sciences Communications*, 11(393).

Portfolio Solutions Group (2024). Can machines build better stock portfolios? the virtue of complexity in the cross-section of stocks. Alternative thinking 2024, issue 4, AQR Capital Management, LLC, Greenwich, CT.

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., and Hassabis, D. (2020). Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44.

TradingView (2025). Gsci index performance chart. Screenshot taken 26 Jul 2025.

van Hasselt, H., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, pages 2094–2100.

Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge.

Zaremba, W., Sutskever, I., and Vinyals, O. (2014). Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Zuckerman, G. (2019). *The Man Who Solved the Market: How Jim Simons Launched the Quant Revolution*. Portfolio, New York.

A. Appendix

A.1 Lookback Windows and Roll Timing Configuration

Table A.1: Commodity-Specific Lookback Windows and Roll Timing Parameters

Commodity	**Active Months**	**Typical Roll Timing**	**Lookback Window (days)**
Crude Oil (WTI)	All months	5-7 days before expiry	20
Brent Oil	All months	5-7 days before expiry	20
Sugar #11	Mar, May, Jul, Oct	1-2 weeks before expiry	50
Silver	Mar, May, Jul, Sep, Dec	2 weeks before expiry	50
Gold	All months	2 weeks before expiry	50
Live Cattle	Feb, Apr, Jun, Aug, Oct, Dec	2-3 weeks before notice day	80

Table A.1 presents the lookback windows used for each commodity. These were calibrated based on each commodity's trading patterns observed from direct experience in futures markets.

A.2 LSTM Network Architecture and Implementation Details

This section provides the complete mathematical specification of the LSTM network architecture used in the LSTM-DDQN trading system. The LSTM component processes sequential market data to capture temporal dependencies in futures price movements and calendar spread patterns.

A.2.1 Input Variables

Table A.2: LSTM Input Variables

Variable	Name	Description	Shape/Range
x_t	Current Input	Input vector at time step t	[input_dim]
h_{t-1}	Previous Hidden State	Output from previous time step	[hidden_dim]
C_{t-1}	Previous Cell State	Long-term memory from previous time step	[hidden_dim]

Table A.2 defines the input variables that feed into the LSTM network at each time step. The current input x_t contains the market features, while the hidden state h_{t-1} and cell state C_{t-1} carry forward information from previous time steps, enabling the network to maintain memory of past market conditions.

A.2.2 Learnable Parameters

Table A.3: LSTM Learnable Parameters

Variable	Name	Description	Shape
W_f	Forget Gate Weights	Weight matrix for forget gate	[hidden_dim, hidden_dim + input_dim]
W_i	Input Gate Weights	Weight matrix for input gate	[hidden_dim, hidden_dim + input_dim]
W_C	Candidate Gate Weights	Weight matrix for candidate values	[hidden_dim, hidden_dim + input_dim]
W_o	Output Gate Weights	Weight matrix for output gate	[hidden_dim, hidden_dim + input_dim]
b_f	Forget Gate Bias	Bias vector for forget gate	[hidden_dim]
b_i	Input Gate Bias	Bias vector for input gate	[hidden_dim]
b_C	Candidate Gate Bias	Bias vector for candidate values	[hidden_dim]
b_o	Output Gate Bias	Bias vector for output gate	[hidden_dim]

Table A.3 lists all trainable parameters in the LSTM architecture. Each gate has its own weight matrix and bias vector, allowing the network to learn distinct behaviors for forgetting old information, incorporating new information, and controlling output. These parameters are optimised during training through backpropagation to minimise the trading loss function.

A.2.3 Intermediate Computations

Table A.4: LSTM Intermediate Computations

Variable	Name	Description	Range	Purpose
f_t	Forget Gate Output	Controls what to forget from cell state	[0, 1]	Memory erasure control
i_t	Input Gate Output	Controls what new info to store	[0, 1]	New memory selection
$\tilde{C}_t$	Candidate Values	New information candidates	[-1, 1]	Potential new memories
o_t	Output Gate Output	Controls what to output from cell state	[0, 1]	Output selection

Table A.4 shows the intermediate computations performed by each LSTM gate. The sigmoid activation functions in the forget, input, and output gates produce values in [0,1], acting as filters that determine information flow. The candidate values use tanh activation to generate new memory content in the range [-1,1].

A.2.4 State Variables

Table A.5: LSTM State Variables

Variable	Name	Description	Range	Role
C_t	Current Cell State	Updated long-term memory	$[-\infty, \infty]$	Internal memory storage
h_t	Current Hidden State	Current output and next input	[-1, 1]	External output

Table A.5 defines the two state variables that carry information forward in the LSTM network. The cell state C_t maintains long-term memory without bounds, while the hidden state h_t serves as both the current timestep output and input to the next timestep, bounded by the tanh activation function.

A.2.5 LSTM Logic Flow

Table A.6: LSTM Logic Flow and Operations

Step	Gate/Operation	Formula	Purpose
1	Forget Gate	$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$	Decide what to forget
2	Input Gate	$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$	Decide what to remember
3	Candidate Values	$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$	Generate new memories
4	Cell State Update	$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$	Update long-term memory
5	Output Gate	$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$	Decide what to output
6	Hidden State	$h_t = o_t * \tanh(C_t)$	Generate final output

Table A.6 presents the sequential computation flow within each LSTM cell. The operations follow a specific order: first determining what information to discard (forget gate), then what new information to store (input gate and candidates), updating the cell state, and finally generating the output. This systematic approach enables the LSTM to selectively retain relevant market information while discarding outdated patterns.

A.3 Hyperparameter Configuration

This section details the complete hyperparameter configuration used in the LSTM-DDQN trading system. These parameters control various aspects of the learning process, risk management, and market simulation, calibrated through systematic experimentation to balance exploration, exploitation, and stable convergence.

Table A.7: Complete Hyperparameter Configuration

Parameter Category	**Parameter**	**Value**
Market Parameters	TICK_VALUE	10.0000
	MAX_CONTRACTS	8
	TRANSACTION_COST_PER_CONTRACT	0.7500
	BID_ASK_SLIPPAGE	0.5000 ticks
Risk Management	MAX_DAILY_LOSS	500.0
	STOP_LOSS_PERCENTAGE	0.3000
Data Parameters	LOOKBACK_WINDOW	20
	TRAIN_RATIO	0.7000
	VAL_RATIO	0.1000
	TEST_RATIO	0.2000
LSTM Parameters	HIDDEN_SIZE	32
	NUM_LAYERS	2
	DROPOUT	0.0500
	BIDIRECTIONAL	False
	PROCESSING_DIM	32
DDQN Parameters	LEARNING_RATE	0.0001
	GAMMA	0.9900
	TAU	0.0050
	BATCH_SIZE	16
Training Parameters	EPISODES	2000
	MEMORY_SIZE	500
	UPDATE_FREQUENCY	2
	TARGET_UPDATE_FREQUENCY	200
	EPSILON_START	1.0000
	EPSILON_DECAY	0.9950

Table A.7 presents the complete hyperparameter configuration organised by functional category. Market parameters reflect realistic trading conditions, risk management settings provide downside protection, and training parameters are tuned to ensure stable convergence while preventing overfitting. The LSTM-DDQN parameters follow established best practices from reinforcement learning literature, while LSTM parameters are sized to capture temporal patterns without excessive computational overhead.

A.4 Experimental Results

A.4.1 Statistical Performance Analysis

Table A.8 presents the 95% confidence intervals for key performance metrics across all commodities, calculated from five independent runs with different random seeds. The confidence intervals provide insight into the statistical reliability of the reported results and the inherent variability in the LSTM-DDQN trading system's performance.

Table A.8: LSTM-DDQN Performance Metrics with 95% Confidence Intervals

Commodity	**Total Return (%)**	**Sharpe Ratio**	**Max Drawdown (%)**	**Win Rate (%)**
Crude Oil	-29.72 ± 0.65 $[-30.37, -29.07]$	-6.67 ± 2.55 $[-9.22, -4.12]$	30.07 ± 0.08 $[29.99, 30.15]$	42.4 ± 15.1 $[27.3, 57.5]$
Brent Oil	-29.88 ± 0.59 $[-30.47, -29.29]$	-4.44 ± 2.34 $[-6.78, -2.10]$	30.04 ± 0.06 $[29.98, 30.10]$	46.8 ± 15.1 $[31.7, 61.9]$
Gold	-7.47 ± 10.72 $[-18.19, 3.25]$	-0.17 ± 0.16 $[-0.33, -0.01]$	12.60 ± 15.88 $[-3.28, 28.48]$	66.6 ± 14.7 $[51.9, 81.3]$
Live Cattle	-5.95 ± 17.44 $[-23.39, 11.49]$	-0.31 ± 1.49 $[-1.80, 1.18]$	7.36 ± 18.48 $[-11.12, 25.84]$	58.9 ± 25.7 $[33.2, 84.6]$
Silver	-30.19 ± 0.29 $[-30.48, -29.90]$	-11.62 ± 7.20 $[-18.82, -4.42]$	30.19 ± 0.29 $[29.90, 30.48]$	50.4 ± 14.1 $[36.3, 64.5]$
Sugar #11	-30.26 ± 0.25 $[-30.51, -30.01]$	-11.75 ± 16.12 $[-27.87, 4.37]$	30.26 ± 0.25 $[30.01, 30.51]$	44.8 ± 19.0 $[25.8, 63.8]$

A.4.2 Statistical Significance Testing

To evaluate the statistical significance of the performance differences between the LSTM-DDQN trading system and the Goldman Roll benchmark strategy, paired t-tests on the return distributions across commodities are conducted. The null hypothesis (H_0) states that there is no significant difference in mean returns between the two strategies, while the alternative hypothesis (H_1) posits that the strategies produce significantly different returns.

Given the limited sample size of six commodities, a two-tailed paired t-test with $\alpha = 0.05$ significance level is employed. The test statistics reveal a mean difference of -18.43% in favor Mou's 2011 Goldman Roll strategy, with a standard error of 3.21%. The calculated t-statistic is $t = -5.74$ with $df = 5$ degrees of freedom, yielding a p-value of $p < 0.005$, indicating statistical significance at the 5% level.

However, several important caveats must be considered when interpreting these results. The test period (2022-2025) coincides with a period when traditional calendar spread arbitrage opportunities have largely been arbitraged away, as documented by Irwin et al. (2022). This temporal factor may systematically disadvantage both strategies relative to historical performance periods.

Furthermore, the high variability observed in Live Cattle and Gold performance metrics (reflected in their wide confidence intervals) suggests that strategy effectiveness is highly commodity-dependent, potentially related to market microstructure differences, liquidity conditions, and the specific characteristics of roll periods across different futures contracts. The consistently narrow confidence intervals for energy commodities (Crude Oil, Brent Oil) and broad intervals for agricultural and precious metals indicate that the LSTM-DDQN system's performance stability varies significantly across asset classes.

A.4.3 Goldman Roll Benchmark Strategy Results

Table A.9: Goldman Roll Strategy Performance (10-day front-run)

Commodity	**Return (%)**	**Total P&L ($)**	**Sharpe Ratio**	**Max Drawdown (%)**	**Win Rate (%)**	**Trades #**	**Profit Factor**
Crude Oil	−4.47	−233.95	−5.10	4.47	0.0	37	0.00
Brent Oil	−4.61	−248.65	−5.20	4.61	0.0	37	0.00
Gold	−1.02	−110.25	−0.83	1.24	45.9	37	2.10
Live Cattle	−4.39	−226.50	−1.88	4.39	45.9	37	0.43
Silver	−4.23	−210.75	−5.02	4.23	0.0	37	0.00
Sugar #11	−4.19	−206.35	−4.91	4.19	0.0	37	0.00
Average	**−3.82**	**−202.49**	**−3.82**	**3.86**	**15.3**	**37**	**0.42**

The Goldman Roll benchmark strategy demonstrates consistently poor performance across all commodities during the test period (2022-2025), with an average return of -3.82% and universally negative Sharpe ratios. Four of the six commodities (Crude Oil, Brent Oil, Silver, Sugar #11) recorded zero winning trades, indicating that the traditional calendar spread arbitrage opportunities identified in earlier literature have largely disappeared.

A.4.4 Detailed Individual Run Results

This section presents the complete performance metrics for each individual training run across all commodities, demonstrating the consistency and variability of the LSTM-DDQN trading system across different random seeds and providing transparency into the statistical robustness of the reported results.

Crude Oil (CL) Individual Run Performance

Table A.10: Crude Oil (CL) Individual Run Results

Run	Seed	Return (%)	Sharpe	Max Drawdown (%)	Win Rate (%)	Trades
1	42	−30.11	−5.83	30.11	57.6	33
2	123	−30.13	−9.23	30.13	38.4	73
3	456	−29.44	−5.93	30.03	37.5	32
4	789	−29.48	−6.42	30.07	41.0	39
5	999	−29.44	−5.92	30.03	37.5	32

Brent Oil (CO) Individual Run Performance

Table A.11: Brent Oil (CO) Individual Run Results

Run	Seed	Return (%)	Sharpe	Max Drawdown (%)	Win Rate (%)	Trades
1	42	−30.02	−4.29	30.02	33.8	77
2	123	−30.04	−3.56	30.04	47.8	46
3	456	−30.04	−3.82	30.04	55.2	58
4	789	−29.27	−3.77	30.11	53.5	43
5	999	−30.01	−6.78	30.01	43.6	110

Gold (GC) Individual Run Performance

Table A.12: Gold (GC) Individual Run Results

Run	Seed	Return (%)	Sharpe	Max Drawdown (%)	Win Rate (%)	Trades
1	42	−15.94	−0.27	26.27	63.1	65
2	123	−2.20	−0.09	6.33	72.7	11
3	456	−12.29	−0.26	15.43	52.6	38
4	789	−4.93	−0.16	9.05	72.2	18
5	999	−2.20	−0.09	6.33	72.7	11

Live Cattle (LC) Individual Run Performance

Table A.13: Live Cattle (LC) Individual Run Results

Run	Seed	Return (%)	Sharpe	Max Drawdown (%)	Win Rate (%)	Trades
1	42	−3.50	−0.62	4.62	45.5	11
2	123	−3.95	−0.57	5.18	53.8	13
3	456	2.40	0.94	0.40	66.7	4
4	789	−23.69	−1.32	26.07	48.6	37
5	999	0.01	0.01	1.50	80.0	5

Silver (SI) Individual Run Performance

Table A.14: Silver (SI) Individual Run Results

Run	Seed	Return (%)	Sharpe	Max Drawdown (%)	Win Rate (%)	Trades
1	42	−30.36	−9.13	30.36	57.6	33
2	123	−30.08	−12.05	30.08	37.9	58
3	456	−30.36	−9.07	30.36	55.6	36
4	789	−30.13	−18.61	30.13	46.8	77
5	999	−30.01	−9.25	30.01	54.3	35

Sugar #11 (SB) Individual Run Performance

Table A.15: Sugar #11 (SB) Individual Run Results

Run	Seed	Return (%)	Sharpe	Max Drawdown (%)	Win Rate (%)	Trades
1	42	−30.06	−9.28	30.06	52.8	53
2	123	−30.36	−7.08	30.36	48.5	33
3	456	−30.36	−7.20	30.36	48.5	33
4	789	−30.36	−7.08	30.36	48.5	33
5	999	−30.17	−28.10	30.17	25.7	101

A.5 Feature Set Specification

Table A.16 details the complete feature vector used as input to the LSTM network, comprising 25 variables that capture price dynamics, liquidity conditions, and inter-contract relationships essential for calendar spread prediction in futures markets.

Table A.16: Complete Feature Set for Model Training

Feature Category	**Features**
Front Month Contract	PX_OPEN1, PX_HIGH1, PX_LOW1, PX_LAST1, PX_VOLUME1, OPEN_INT1
Deferred Contract	PX_OPEN2, PX_HIGH2, PX_LOW2, PX_LAST2, PX_VOLUME2, OPEN_INT2
Volume Metrics	VOL Change1, Vol Change %1, VOL Change2, Vol Change %2, Vol Ratio, Vol Ratio Change
Open Interest Metrics	OI Change1, OI Change %1, OI Change2, OI Change %2, OI Ratio, OI Ratio Change
Calendar Spread	CALENDAR (target variable)