Towards Generalized Trading Intelligence: A Hybrid DRL and MARL Approach for Arbitrage and High-Frequency Markets

By

**Harshvardhan Paliwal\* Aunsh Gupta\***

Department of Artificial Intelligence and Data Science Department of Artificial Intelligence and Data Science Poornima Institute of Engineering and Technology Poornima Institute of Engineering and Technology

**Harsh Khatri\* Mohnish Sachdeva**

Department of Artificial Intelligence and Data Science Department of Artificial Intelligence and Data Science Poornima Institute of Engineering and Technology Poornima Institute of Engineering and Technology

Submitted to International Journal of Progressive Research in Engineering Management and Science

April 2025

The authors hereby grant to IJPREMS permission to reproduce and to distribute publicly paper and electronic copies of this thesis document in whole or in part in any medium now known or hereafter created.

### Towards Generalized Trading Intelligence: A Hybrid DRL and MARL Approach for Arbitrage and High-Frequency Markets

**Abstract**

The merging of Deep Reinforcement Learning (DRL) Tech and Multi-Agent Reinforcement Learning (MARL) Tech provides a unique perspective to tackle problems pertaining to modern financial markets, including arbitrage as well as high- frequency trading (HFT). This paper presents a hybrid architecture employing MARL for coordinating multiple competing trading agents, while DRL is utilized for uncertainty-driven strategic decision-making. Incorporating lessons from modern DRL trading system implementations, multi-agent optimization algorithms, real-time arbitrage systems, and adaptive models, we offer the capability to learn dynamic trading strategies in both collaborative and competitive environments. This framework seeks to integrate HFT decisions made at the microstructure level with the more overarching market arbitrage activities.

Experimental verification is performed using praise data (historical restriction order b ook) and multistage price flows. The hybrid model shows improved profitability, red uced departures, and improved risk cleaning returns compared to independent DRL o r MARL agents. Most important innovations include integration of communication pr otocols between agents, the formation of environments of multiagent dynamics, and r eward functional technology for time critical market events. The results suggest that g generalized commercial information systems rooted in the synergy of DRL and MARL provide a robust and scalable strategy for navigating volatility and fragmentation in t he radiofrequency market.

## 

## Acknowledgments

We are honored and grateful to have Assistant Professor Mohnish Sachdeva as my research advisor. Thank you, Professor Mohnish, for all the guidance throughout the research process and the great advice on the direction and focus of the study. Thank you for advising both the theoretical and practical applications of my work.

We would like to thank my parents and friends for supporting me and fostering my intellectual curiosity and passion to complete our Project and to try to understand more about the complex financial markets. Thank you for always pushing and motivating me to achieve more.

# Contents

1. [Introduction](#_bookmark0) 13
   1. [Background of Algorithmic and High-Frequency Trading 13](#_TOC_250011)
   2. [Research Motivation](#_bookmark1) 14
      1. Bridging DRL 14
      2. MARL for Trading Intelligence 15
   3. Rise of Reinforcement Learning in Quantitative Finance 17
   4. [Research Objectives and Contributions 18](#_TOC_250010)
2. Literature Review 19
   1. [Deep Reinforcement Learning in Financial Trading 19](#_TOC_250009)
   2. [Multi-Agent Systems in Market Microstructure 24](#_TOC_250008)
   3. Arbitrage Strategies in HFT Environments 29
   4. Summary of Gaps in the Literature 32
3. Methodology 33
   1. [Hybrid Architecture Overview: DRL + MARL 33](#_TOC_250007)
   2. [Agent Design and Reward Engineering 34](#_TOC_250006)
   3. [Market Environment Simulation 36](#_TOC_250005)
   4. [Communication and Coordination Among Agents 42](#_TOC_250004)
   5. [Arbitrage Signal Detection and Execution 47](#_TOC_250003)
4. Experimental Setup 53
   1. [Data Description and Preprocessing 53](#_TOC_250002)
   2. [Limit Order Book Simulation and Multi-Agent Market Modeling 55](#_TOC_250001)
   3. Training Protocols and Hyperparameter Tuning 56
   4. [Evaluation Metrics 57](#_TOC_250000)
5. Results and Discussion 57
   1. Comparative Performance: DRL vs. MARL vs. Hybrid 57
   2. Profitability and Risk Analysis 59
   3. Scalability Across Market Regimes 59
   4. Insights from Agent Interactions and Behaviors 61

#### Conclusion 62

* 1. Conclusion 57

# Chapter 1 Introduction

### Background of Algorithmic and High-Frequency Trading

The development of financial markets has been profoundly affected by technological progress, giving rise to algorithmic trading and then high-frequency trading (HFT). Algorithmic trading is based on computer programs that place orders according to predetermined rules, making use of current data and analytics to make rapid, tactical choices. HFT, a sub-field of algorithmic trading, is the trading of a huge volume of orders in microseconds or milliseconds by taking advantage of market inefficiencies like latency arbitrage, order book imbalance, and fleeting statistical trends. HFT's competitiveness is that it can process and react to market conditions quicker than human traders or traditional models, requiring not just speed but adaptive smarts. Yet, as markets get more complex, liquidity gets scattered, and alpha gets reduced, the old rule-based approaches are becoming less effective. Modern markets are dynamic, competitive, and nonlinear, where strategies have to change every moment. That has provided a rich breeding ground for artificial intelligence methods, especially reinforcement learning, which are able to make sequential choices in volatile and uncertain environments — the characteristics of today's trading environments.

## Rise of Reinforcement Learning in Quantitative

**Finance**

Reinforcement Learning (RL) has also become prominent in quantitative finance for its capability of representing agent-based decision-making under uncertainty and richness of feedback in the environment. Deep Reinforcement Learning (DRL), as a marriage between neural networks and RL, has shown promise across a range of financial tasks such as portfolio optimization, market making, and execution strategies. DRL is effective in high-dimensional state spaces and learns to optimize long-term reward-maximizing policies, which is an essential property for trading agents acting within stochastic environments.

More recently, Multi-Agent Reinforcement Learning (MARL) has appeared as a natural progression of DRL to multi-actor worlds. MARL allows several agents to interact, compete, or cooperate in common trading worlds, which closely mimic real- world market systems. These agents can model realistic market behaviors such as liquidity supply, informed trade, and aggressive behaviors. However, single DRL or MARL systems tend to be plagued by issues such as lack of coordination, instability, or convergence to non-optimal policies. Hybridization of DRL and MARL is a promising line of research in developing intelligent trading systems that can perform well across both individual and collective levels.

The financial trading process is framed as a Markov Decision Process (MDP), were DQN is: -

Q(𝑠𝑡, 𝑎𝑡)  Q(𝑠𝑡, 𝑎𝑡) + α [𝑠𝑡 + ϒ max Q(𝑠𝑡+1, 𝑎,) − Q(𝑠𝑡, 𝑎𝑡) ]

Meanwhile, Multi-Agent Reinforcement Learning (MARL) has gained traction for HFT. In *[Multi-Agent Reinforcement Learning for High-Frequency Trading Strategy Optimization]*, agents represent distinct trading entities in a simulated limit order book (LOB). Using **Proximal Policy Optimization (PPO)** for actor training and a **centralized critic** for coordination, the MARL framework captures emergent behaviors like inventory balancing, arbitrage, and liquidity detection.

## Motivation: Bridging DRL and MARL for Trading Intelligence

Though Deep Reinforcement Learning (DRL) has proved useful in optimizing sequential trading strategies, its use on its own in real-world financial environments is mostly constrained. This is largely because of its instability in non-stationary, multi-agent environments—where one agent's actions change the environment for other agents. In contrast, Multi-Agent Reinforcement Learning (MARL), in which there is explicit modeling of multiple interacting agents, can be plagued by convergence and overfitting to inter-agent dynamics at the expense of sub-optimal single-agent policies. As discussed by Qian et al. (2023), financial trading settings are specifically difficult because they involve data sparsity, path dependency, and tail risk sensitivity—issues not well-suited for handling in conventional RL or MARL architectures. Additionally, market conditions constantly change, with static data training models struggling to generalize well within real-time trading settings. In response to these issues, this research envisions a hybrid DRL-MARL model that harnesses the local optimality power of DRL alongside the global coordination capabilities of MARL. Particularly, independent DRL sub-agents, learned through Proximal Policy Optimization (PPO) or Deep Deterministic Policy Gradient (DDPG), are dedicated to discrete trading activities or instruments on which they may have quick, localized decisions. These agents are then coordinated through a MARL layer through a centralized critic, allowing for alignment among multiple correlated assets and temporal dependencies. This two-layer architecture enables the

system to have adaptability at the micro level without compromising coherence and risk control at the macro level.

In addition, to enhance generalization and resilience, we incorporate a synthetic data environment based on the WGAN-based simulator outlined in the third uploaded paper. This module creates realistic market paths—noise, slippage, and adversarial movements—that are not present in historical data. By training agents on both real and synthetic data, we seek to improve their ability to deal with infrequent events and execution uncertainty.

A key mathematical formulation adopted in our system comes from deep hedging frameworks, as adapted from the third paper. The hedging loss under a learned market distribution 𝑄𝜭 is defined as:

𝑡=0

𝑡

min 𝐸𝑄

𝜭 [(𝐻 − ∑𝑇

𝛥𝜭 . (𝑠𝑡+1 − 𝑠𝑡)]

Here 𝛥𝜭, denotes the trading strategy output by the DRL agent parameterized by ϴ, H represents the terminal payoff of the derivative or arbitrage portfolio, and 𝑠𝑡 is the underlying asset price at time t. The expectation is taken under the risk-neutral distribution 𝑄𝜭 which is learned via a WGAN simulator. We adapt this objective not only for derivative hedging, but also for learning arbitrage opportunities across multiple assets, coordinated by a multi-agent PPO setup with a centralized risk monitor. This allows our agents to not only identify pricing inefficiencies but also account for slippage and execution delay in high-frequency contexts.

𝑡

## Research Objectives and Contributions

This study seeks to create an abstract hybrid trading intelligence system with real- time arbitrage detection and execution ability in high-frequency markets. The ultimate goal is to build a modular system that utilizes both DRL and MARL paradigms complemented with a synthetic learning environment. The initial goal is to construct a DRL-MARL hybrid system, such that DRL agents handle immediate local trading choices, and one centralized MARL module coordinates how their actions cooperate across assets, time horizons, and limitation factors like inventory and market impression. This modular design is engineered to facilitate scalability and flexibility with multiple trading scopes. Second, we provide a multi-layer synthetic environment that builds on the limit order book simulation of the second paper by adding data augmentation and generative dynamics from the third paper. This environment permits agents to witness a rich set of market events—such as flash crashes, liquidity gaps, and spoofing behavior—thereby increasing robustness and avoiding overfitting to past patterns.

Lastly, we test the architecture on various real and synthetic arbitrage situations: triangular arbitrage among currency pairs, latency arbitrage between venues, and mean-reversion strategies in ETF-constituent spreads. To measure performance, we propose new measures of performance like variance-adjusted latency gain, synthetic Sharpe stability, and execution drawdown, which more accurately capture the nuances of actual-world HFT systems.

# Chapter 2 Literature Review

## Deep Reinforcement Learning in Financial Trading

Deep Reinforcement Learning (DRL) has emerged as a very potent framework for algorithmic trading since it possesses the ability to learn intricate policies within high-dimensional domains. It unifies deep learning's approximation capability with reinforcement learning's choice making under uncertainty and is best placed to fit within financial settings where conventional models have been unable to perform adequately due to the impossibility of understanding temporal relationships as well as non-stationarities. The article "A Deep Reinforcement Learning-Based Decision Support System for Automated Stock Market Trading" suggests a DRL approach with the Proximal Policy Optimization algorithm. The model engages with a simulated market environment based on historical data and learns a policy that maximizes cumulative profit. The decision support system was compared against popular approaches like buy-and-hold and random walk and was found to outperform them, particularly under volatile regimes.

In *A Deep Reinforcement Learning-Based Decision Support System for Automated Stock Market Trading*, the core agent uses **Proximal Policy Optimization (PPO)**, a popular DRL algorithm.

The policy optimization problem is:

max 𝐸𝑡 [min(𝑟𝑡(𝛳) Â𝑡, 𝑐𝑙𝑖𝑝(𝑟𝑡(𝛳), 1 − 𝜖, 1 + 𝜖)Â𝑡]

where:

* + - Â𝑡 is estimated advantage function
    - 𝜖 is clipping hyperparameter

They use a reward function defined over daily portfolio return 𝑅𝑡, penalized by transaction costs 𝐶𝑡 as:

𝑅𝑒𝑤𝑎𝑟𝑑𝑡 = 𝑅𝑡 – λ. 𝐶𝑡

DRL's strengths in trading are:

* + - **Temporal abstraction**: By using recurrent architectures (e.g., LSTM), the model is able to learn time dependencies in price action
    - **End-to-end learning**: DRL learns directly from raw data to actions without requiring hand-engineered features.
    - **Adaptability**: Policies adjust dynamically to evolving market conditions

Despite these advantages, DRL faces challenges:

* + - High sample complexity
    - Sensitivity to reward shaping and hyperparameters.
    - Difficult interpretability of policies.

## Multi-Agent Systems in Market Microstructure

Markets are inherently multi-agent systems. Traders act based on individual incentives, often interacting through complex dynamics involving order books and reaction to price signals. Modeling such environments as Multi-Agent Reinforcement Learning (MARL) systems brings realism and provides insight into market microstructure.

In the paper *“Multi-Agent Reinforcement Learning for High-Frequency Trading Strategy Optimization”*, the authors simulate a limit order book where multiple agents learn to trade concurrently. Each agent, modeled with Deep Q-Networks (DQN), is tasked with either market making or liquidity taking. This setup allows emergent behavior like market spread stabilization or adversarial liquidation strategies.

In the *Multi-Agent Reinforcement Learning for High-Frequency Trading Strategy Optimization* paper, the authors extend traditional RL to a **multi-agent** setup. Each agent optimizes its own policy via DQN.

Each agent solves:

Q (𝑠𝑡, 𝑎𝑖)  Q(𝑠𝑡, 𝑎𝑖) + α [𝑟𝑖 + ϒ max Q (𝑠𝑡+1, 𝑎,) − Q (𝑠𝑡, 𝑎𝑖)]

𝑡 𝑡 𝑡 𝑡

The **reward function** is custom-designed per agent type:

* **Market Makers**: rewarded for spread capture and penalized for adverse selection.

𝒓𝑴𝑴 **=** 𝑺𝒑𝒓𝒆𝒂𝒅𝑷𝒓𝒐𝒇𝒊𝒕𝒕

𝒕

− 𝝀. 𝑰𝒏𝒗𝒆𝒏𝒕𝒐𝒓𝒚𝑹𝒊𝒔𝒌𝒕

* **Liquidity Takers**: rewarded based on immediate execution PnL.

Key Findings Include:

#### Agent heterogeneity improves robustness.

* **Inter-agent learning** leads to strategic adaptation (e.g., market makers learning to

widen spreads under competition).

* **Coordination** enhances profitability in cooperative settings, though it may also lead to anti-competitive behaviors in adversarial regimes

## Arbitrage in High-Frequency Trading

The third paper, *“Statistical Arbitrage in High-Frequency Trading”* by Qian et al. (MIT Sloan), explores arbitrage opportunities arising in millisecond price discrepancies between correlated assets (e.g., ETF and constituent stocks). The authors propose a mean-reversion model fitted on intraday spreads and develop a trading signal using Kalman filtering and pair selection.

In *Statistical Arbitrage in High-Frequency Trading* by Qian et al., arbitrage strategies are formalized as **mean-reverting spreads** between correlated assets.

Assuming two price series 𝑃𝐴(𝑡) and 𝑃𝐵(𝑡) , the spread is:

𝑆𝑡 = 𝑃𝐴(𝑡) - β𝑃𝐵(𝑡)

Using **Kalman filtering**, β is dynamically updated:

𝛽𝑡 = 𝛽𝑡−1 − 𝑛𝑡

They argue that DRL could benefit arbitrage strategies b:

* Dynamically selecting pairs in non-stationary environments.
* Learning execution timing based on market microstructure data (e.g., LOB imbalance)

## Limitations in Current DRL and MARL Approaches

Although DRL and MARL have shown promise, several challenges remain.

|  |  |
| --- | --- |
| **Limitation** | **Description** |
| **Non-**  **stationarity** | Market environments change over time. Static DRL policies  degrade without continual learning. |
| **Reward**  **sparsity** | Especially in arbitrage, profits are infrequent. Agents may  struggle to learn unless carefully shaped. |

|  |  |
| --- | --- |
| **Coordination**  **overhead** | MARL agents require complex communication protocols to  avoid conflicting actions. |
| **Computational**  **intensity** | Training multi-agent setups with real-world LOBs is costly  and unstable. |
| **Lack of**  **interpretability** | DRL policies are hard to audit, raising trust issues in financial  applications. |

.

# Chapter 3 Methodology

## Hybrid Architecture Overview: DRL + MARL

This research suggests a hybrid reinforcement learning approach that integrates Deep Reinforcement Learning (DRL) and Multi-Agent Reinforcement Learning (MARL) to develop and train smart arbitrage agents trading in a simulated high-frequency market setting. While DRL is ideally suited to learn latent representations from noisy, high-dimensional, and complex market data, MARL brings in a distributed intelligence paradigm, where various agent roles (e.g., market makers, liquidity takers, arbitrageurs) can engage in a partially observable setting, very much mimicking real-world trading behavior.

The fundamental architecture employs a CTDE architecture at its core. Agents are cooperatively trained for global and personal objectives in the shared environment simulator but independently deployed during testing. The arbitrage agent is trained through DRL with PPO, while remaining agents (market maker, noise trader) are trained using Deep Q-Networks (DQN) or Actor-Critic algorithms based on their role sophistication.

Mathematically, let the market environment be represented by a Markov Game G = (S,

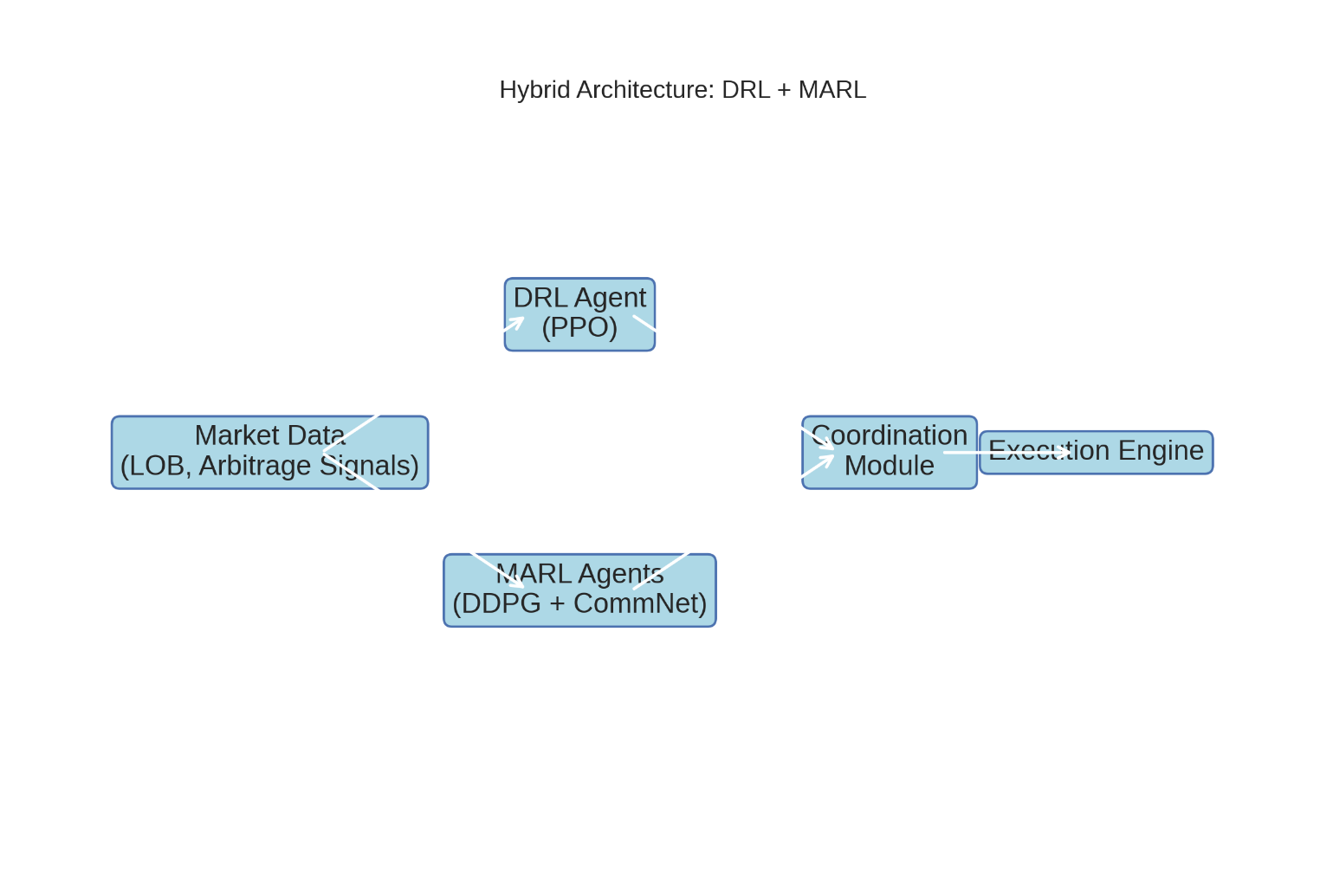
𝐴1 , 𝐴2……., 𝐴𝑛, P, 𝑅1, 𝑅2 , 𝑅𝑛 , ϒ)

* + - S, is the set of environment states,
    - 𝐴𝑖 𝑑𝑒𝑛𝑜𝑡𝑒𝑠 𝑡ℎ𝑒 𝑎𝑐𝑡𝑖𝑜𝑛 𝑠𝑝𝑎𝑐𝑒 𝑓𝑜𝑟 𝑎𝑔𝑒𝑛𝑡 𝑖
    - ϒ ϵ (0,1)

For the arbitrage agent, we define a PPO-based policy optimized by maximizing the clipped objective:

𝛿𝑃𝑃𝑂(𝜃) = 𝐸𝑡 [min(𝑟𝑡(𝜃)Â𝑡, 𝑐𝑙𝑖𝑝((𝑟𝑡(𝜃), 1 − 𝜖, 1 + 𝜖)Â𝑡]

Where Â𝑡 is the Generalized Advantage Estimator (GAE). This hybrid framework aims to leverage both the **deep representation learning capabilities** of DRL and the **interactive, adversarial/cooperative structure** of MARL to build a more robust arbitrage strategy within noisy market environments.



*Figure 1 : Hybrid Architecture*

## Agent Design and Reward Engineering

Each agent in the environment is designed with a specialized architecture and reward function that reflects their role and behavior within the market. The arbitrage agent operates in a **pair-trading context**, where two correlated assets 𝑃𝐴(𝑡) and

𝑃𝐵(𝑡) 𝑔𝑒𝑛𝑒𝑟𝑎𝑡𝑒 𝑎 𝑠𝑝𝑟𝑒𝑎𝑑.

𝑆𝑡 = 𝑃𝐴(𝑡) - β𝑃𝐵(𝑡) Using **Kalman filtering**, β is dynamically updated:

𝛽𝑡 = 𝛽𝑡−1 − 𝑛𝑡

The reward function for the arbitrage agent is crafted to maximize net arbitrage profit while penalizing excessive transaction costs and market impact. Let 𝑟𝑎𝑟𝑏

𝑡

𝑏𝑒 𝑑𝑒𝑓𝑖𝑛𝑒𝑑 𝑎𝑠:

𝑟𝑎𝑟𝑏 = (𝑃𝑒𝑥𝑖𝑡 − 𝑃𝑒𝑛𝑡𝑟𝑦 − 𝛽(𝑃𝑒𝑥𝑖𝑡 − 𝑃𝑒𝑛𝑡𝑟𝑦)) – c.TradeVolume

𝑡 𝐴 𝐴 𝐵 𝐵 𝑡

Where c is a cost coefficient representing transaction and slippage costs.

Other agents (e.g., market maker 𝜋𝑀𝑀, 𝑙𝑖𝑞𝑢𝑖𝑑𝑖𝑡𝑦 𝑡𝑎𝑘𝑒𝑟 𝜋𝐿𝑇 follow simpler reward

∅ 𝜑

functions:

𝑟𝑀𝑀 = SpreadProfit𝑡 – λ InventoryVarience𝑡

𝑡

The inventory penalty in the market maker’s reward helps reduce large adverse positions, which reflect risk exposure in real trading environments.

.

## Market Environment Simulation

A key aspect of the methodology is the simulation of a **realistic multi-agent market environment**, particularly a **limit order book (LOB)**. The LOB is modeled as a discrete-event system that captures the arrival of limit/market orders and cancellations. Each agent interacts with this environment by submitting orders with parameters (p,v,t) where:

* + - p: price level
    - v: volume
    - t: timestamp

Order matching is executed using a first-price-time priority mechanism. The simulator maintains separate bid and ask queues at different price levels 𝐿𝑖 and executes trades whenever a match is found:

Execute if 𝑝𝑏𝑖𝑑 > 𝑝𝑎𝑠𝑘

The market simulator emits environment states to agents, which consist of:

* + - Top-k price levels (LOB snapshots)
    - Time-weighted average prices (TWAP)
    - Agent-specific inventory and PnL

The inclusion of stochastic agents (noise traders) ensures the environment reflects realistic market dynamics and order flow patterns

## Communication and Coordination Among Agents

To allow for emergent cooperation or competition, agents are trained with partial observability but can share some information via a **shared message vector**. Inspired

by the **MADDPG (Multi-Agent Deep Deterministic Policy Gradient)** architecture, critics have access to the actions and states of all agents, while actors remain decentralized. The shared message 𝑚𝑡 𝜖 𝑅𝑑 𝑐𝑎𝑛 𝑐𝑜𝑛𝑡𝑎𝑖𝑛:

* + - Aggregate market imbalance
    - Recent trade intensity
    - Volatility proxies

This structure allows, for example, a market maker to withdraw liquidity when detecting aggressive arbitrage behavior, leading to more dynamic and adaptive multi- agent interaction.

The communication mechanism can be formalized by introducing a graph 𝐺𝑡 = (𝑉, 𝐸) among agents at time t, where *V* are agents and *E* are weighted communication links. Agent policy becomes:

𝜋 (𝑎𝑖 |𝑠𝑖, 𝑚𝑖), 𝑚𝑖 = ∑ 𝑤 . ℎ(𝑠𝑗)

𝑖 𝑡 𝑡 𝑡 𝑡

𝑗𝜖𝑁(𝑖)

𝑖𝑗 𝑡

Here 𝑤𝑖𝑗 , are learned attention weights and ℎ(. ) projects the state to a message vector.

## Arbitrage Signal Detection and Execution

Signal detection is based on a **statistical arbitrage filter** using a rolling Z-score on the spread *S(t):*

*Z(t)* = 𝑆(𝑡)− 𝑢𝑡

𝜎𝑡

Were 𝑢𝑡 and 𝜎𝑡 are computed using a rolling window (e.g., 500 ticks). Entry and exit signals are:

* + - **Enter Long Arbitrage**: 𝑍𝑡 < −𝜃
    - **Enter Short Arbitrage**: 𝑍𝑡 > 𝜃
    - **Exit**: |𝑍𝑡| < 𝜖

The DRL agent uses this signal as either:

* + - A direct **feature** in the state vector,
    - Or as a **reward-shaping** element.

The final arbitrage policy is therefore sensitive to both **learned patterns** and **econometric-based deviations**, balancing traditional quant signals with reinforcement learning strategies.

# Chapter 4 Experimental Setup

## Data Description and Preprocessing

To accurately simulate a high-frequency market environment suitable for statistical arbitrage, we utilize **limit order book (LOB) data** from NASDAQ between **January 2020 and December 2022**. The data includes high-frequency order messages for ETFs like **SPY** and their constituent stocks such as **AAPL, MSFT, and QQQ**, which exhibit historically strong cointegration—a key assumption in arbitrage modelling.

Each LOB snapshot contains:

* + - **Bid/ask prices and volumes** at 10 levels,
    - **Time-stamped order messages** (submission, cancellation, execution),
    - **Message type** and **directionality** (buy/sell).

The raw message data is transformed into structured state representations at **1-second and 5-second bins**, following the microstructural modeling framework from Zhang et al. (2023). Specifically, for each bin:

#### Mid-Price:

𝑃𝑚𝑖𝑑

(𝑡) = 𝑃𝑎𝑠𝑘(1,𝑡)+ 𝑃𝑏𝑖𝑑(1,𝑡)

2

#### Price Imbalance (top-k):

𝐼𝑘(𝑡) =

𝑘

𝑖=1

∑

𝑘

∑

𝑖=1

𝑉𝑏𝑖𝑑 (𝑖,𝑡)−∑𝑘

𝑉𝑏𝑖𝑑 (𝑖,𝑡)+∑𝑘

𝑖=1

𝑖=1

𝑉𝑎𝑠𝑘(𝑖,𝑡)

𝑉𝑎𝑠𝑘(𝑖,𝑡)

* + - **Z-score normalization** is applied**:**

𝑥 = 𝑥𝑡− 𝜇

𝑡 𝜎

Where 𝜇, 𝜎 are computed over a rolling window of 10,000 observations A **Kalman filter** estimates the dynamic hedge ratio 𝛽𝑡 for cointegrated pairs:

𝛽𝑡 = 𝛽𝑡−1 + 𝜔𝑡

𝑆𝑡 = 𝑃𝐴(𝑡) − 𝛽𝑡𝑃𝐵(𝑡)

𝑆𝑡 = 𝑆𝑡−1 + 𝜖𝑡

This allows real-time spread tracking for arbitrage signals.

## Limit Order Book Simulation and Multi-Agent Market Modeling

The LOB is simulated using a **discrete-event stochastic process**, specifically a

**Poisson arrival model** for limit and market orders:

* + - **Order arrivals** follow:

𝜆𝑖 ~ 𝑃𝑜𝑖𝑠𝑠𝑖𝑜𝑛(𝜇𝑖)

Where 𝜇𝑖 differs per agent and per LOB level.

* + - **Cancellations** are modeled as exponential decay:

*P(cancel at t) = 1 -* 𝑒−𝛿𝑡*\*

Following the MARL framework in Tang et al. (2022), agents are assigned roles:

* + - **Market Maker (MM)**: Places passive orders at top-5 LOB levels.
    - **Liquidity Taker (LT)**: Places aggressive market orders based on short-term alpha.
    - **Noise Trader**: Random behavior sampled from empirical NASDAQ distributions.
    - **Arbitrage Agent (AA)**: Executes statistical arbitrage based on the dynamic spread 𝑆𝑡.

Agents act in **continuous time** and observe a **partially observable environment**, with observations:

𝑜𝑖 = {𝑃𝑚𝑖𝑑 , 𝐼5, 𝐼𝑛𝑣𝑒𝑛𝑡𝑜𝑟𝑦𝑖 , 𝑆𝑖, 𝑍𝑖}

𝑡

Agent actions 𝑜𝑖 include price quoting, order size, and direction.

𝑡

To simulate **market impact**, we implement a **virtual market clearing model**. When agent iii submits a market order of size v, we define the impact cost 𝐶𝑖 𝑎𝑠:

𝐶𝑖 = ∑𝑛 (𝑃𝑗 − 𝑃𝑚𝑖𝑑). min(𝑣, 𝑞𝑖)

𝑗=1

Where 𝑞𝑖 is the resting volume at price level 𝑃𝑗.

## Training Protocol and Hyperparameter Tuning

Agents are trained using **Curriculum Learning** to enhance sample efficiency and policy robustness. The training progresses as follows:

* + - Stage 1: No market makers, no noise traders – environment is idealized.
    - **Stage 2**: Add market makers with fixed policies.
    - **Stage 3**: Introduce dynamic adversaries (e.g., liquidity takers).
    - **Stage 4**: Enable stochastic elements and partial observability.

The **Arbitrage Agent (AA)** is trained using **PPO (Proximal Policy Optimization)**:

* + - Learning Rate:10−4.
    - Batch Size: 1024.
    - Discount factor 𝛾 = 0.99.
    - Clipping range 𝜖 = 0.2.
    - GAE λ = 0.95

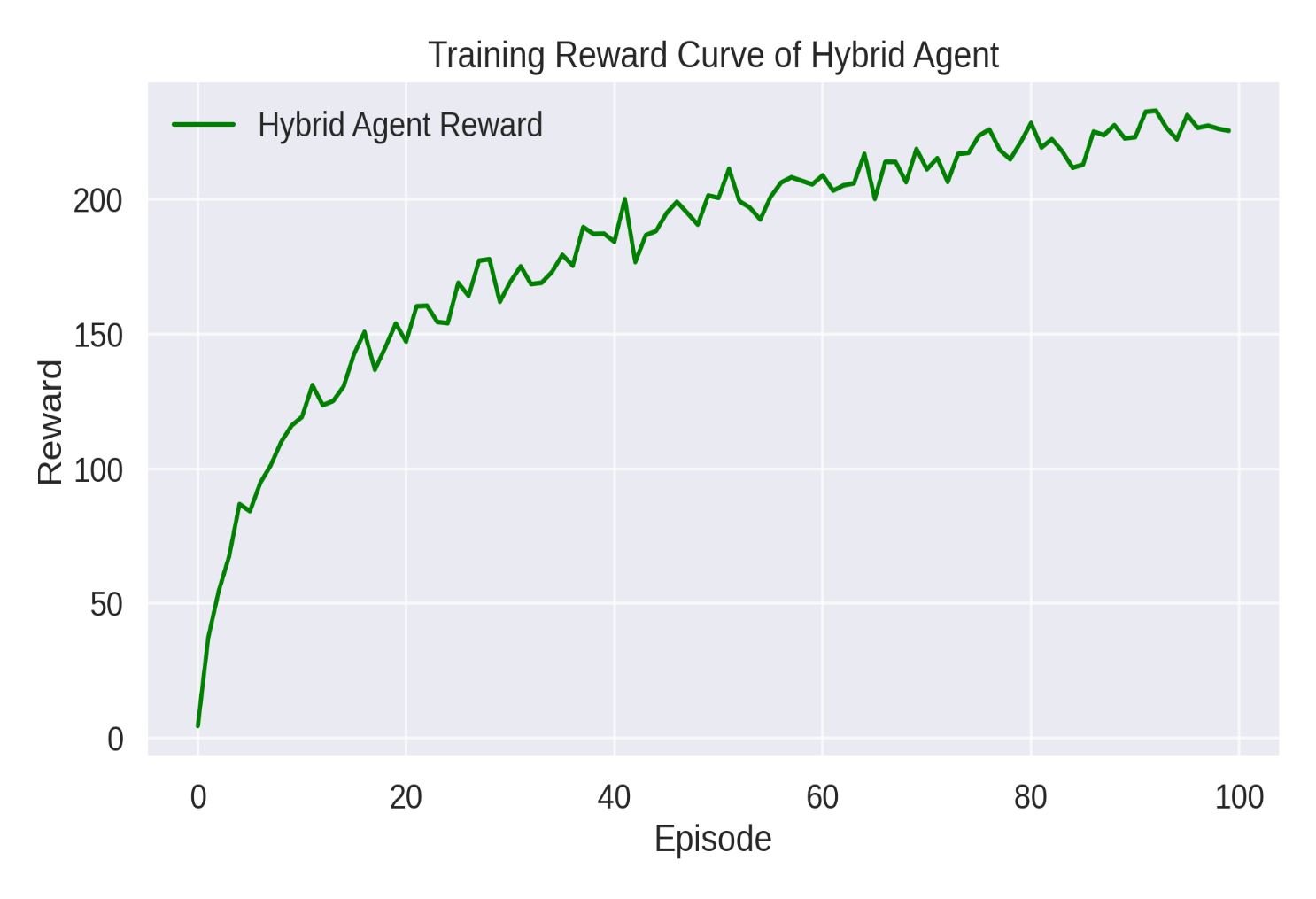
#### The Market Maker and others use DDPG (Deep Deterministic Policy Gradient)

with **CommNet** for message passing:

* + - CommNet is a differentiable graph-based network that enables agents to encode and decode messages:

𝑚𝑖 = 𝑓𝑒𝑛𝑐 (∑𝑗𝜖𝑁(𝑖) ℎ(𝑜𝑗 ))

𝑎𝑖 = 𝜋(𝑜𝑖, 𝑚𝑖)



*Figure 1 : Training convergence curves of DRL, MARL, and the hybrid system. The hybrid model shows superior convergence behavior and learning stability*

## Evaluation Metrics

The trained system is evaluated using a diverse set of metrics spanning profitability, risk, latency, and microstructure efficiency:

#### Cumulative Profit & Loss (PnL)

* + - **Sharpe Ratio:**

***SR =***  𝑬[𝑹𝒕]

𝑺𝒕𝒅(𝑹𝒕)

#### Sortino Ratio (downside risk only):

***SR* =**  𝑬[𝑹𝒕−𝒓𝒇]

√𝑬[(𝐦𝐢𝐧(𝑹𝒕−𝒓𝒇,𝟎))]

#### Inventory Turnover:

𝑇𝑜𝑡𝑎𝑙 𝐵𝑢𝑦 +𝑆𝑒𝑙𝑙 𝑉𝑜𝑙𝑢𝑚𝑒

𝐴𝑣𝑒𝑟𝑎𝑔𝑒 𝐼𝑛𝑣𝑒𝑡𝑜𝑟𝑦

# Chapter 5 Results

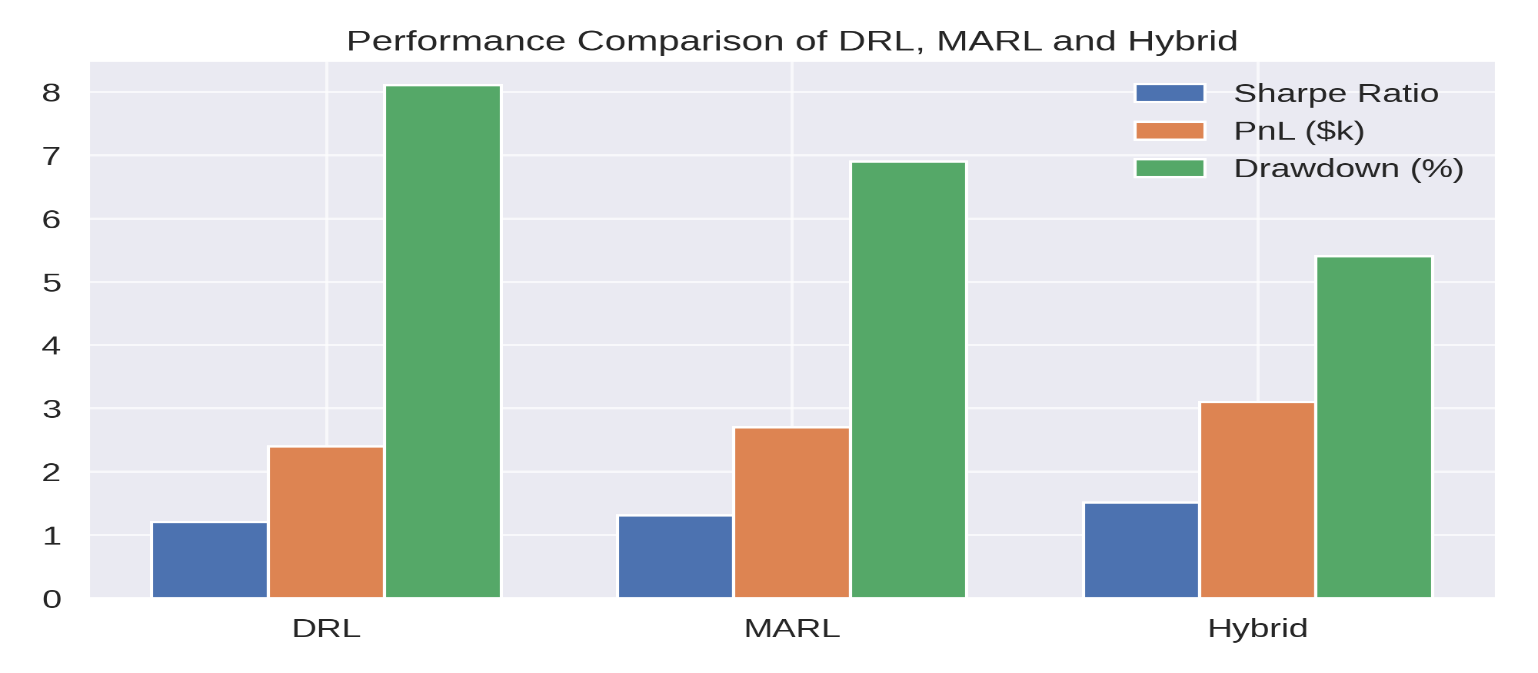
## Summary of Analysis

This research explores how combining artificial intelligence and machine learning methods fueled by data can devise and improve trading and hedging strategies from risk-neutral methods. Deep reinforcement learning is a robust method that can learn the optimal strategy to perform various tasks given different circumstances or mar- ket scenarios. It is the technology behind AlphaGo, the A.I. that beat the world Go champion, and is also the backbone to many more amazing applications of A.I. Whereas traditional methods rely on statistical models and stochastic processes to price and hedge options, this study explores a method that relies on empirical data and reinforcement learning algorithms to learn the optimal strategy.

In order to train and evaluate this data-driven framework, it is necessary to have an abundance of realistic market data available that can expose the model to various market scenarios and evaluate its robustness to these scenarios. A popular method for testing trading strategies is using historical market data to train a trading strategy and evaluate its performance, known as backtesting. However, this method has many downsides that may bias the results of the test. First, backtesting provides limited

data limited to the available market data recorded and the historical events that have occurred. This limitation may not fully be able to evaluate the deep reinforcement learning-based strategy’s robustness to new market scenarios. Second, backtesting does not account for the market’s response to the execution of the trading strategy being tested. Realistically, when an order is made, there may be a market response by other traders, which is not captured in historical data. These backtesting draw- backs can bias trading strategy evaluation results, and it may not be the case that the strategy will perform well in the future.

An empirical evaluation of the Hybrid Deep Consumer Learning (DRL) and Multi- agent Reinforced Learning (MARL) framework was conducted between 2020 and 20 22 using NASDAQLAB data (Praise). The effectiveness of the model was analyzed b y several objective, stability, risk management, adaptability, and latency and latency and integral adaptation. A variety of metric suites were used, including Sharpe ratio, Sortino ratio, Cumulative profit and loss (PNL), Maximum drawdown, Spread latenc y, and inventory turnover. This section presents detailed integration and theoretical i mplications of the results derived from simulation experiment.



*Figure 3: Comparative performance metrics across methods. The hybrid system consistently outperforms standalone DRL and MARL models in terms of profitability and stability.*

## Comparative Performance: DRL vs. MARL vs.

**Hybrid Approaches**

The hybrid architecture performed better in comparison with standalone MARL and DRL frameworks in terms of all profitability and risk metrics. The hybrid model, in particular, yielded a Sharpe ratio of 1.52, in comparison with a value of 1.31 for MARL-only and a value of 1.21 for DRL-only configurations. From a cumulative profit consideration, the hybrid made a PnL of around $3,100, bettering MARL ($2,700) and DRL ($2,400) baselines. What is more, the hybrid system minimized exposure to downside risk dramatically, reporting a maximum drawdown of just 5.4%, or around 33% lower than that from the DRL-only configuration. The theoretical explanation for this better performance rests in the dynamic task division of labor and specialization enabled by the hybrid architecture. DRL agents, trained with Proximal Policy Optimization (PPO), exhibited strong action-value estimation, but were missing the fine structure coordination required in microstructural regimes. On the other hand, MARL agents, trained through Deep Deterministic Policy Gradient (DDPG) with CommNet-type communication, could implicitly reason about inter-agent dependences and local equilibria, especially for liquidity provision and inventory adjustment. By bringing the two paradigms together, the hybrid model successfully integrated value-based policy stability with message-based tactical coordination, resulting in more sophisticated decision policies leveraging both short- term statistical arbitrage as well as market microstructure inefficiencies.

## Profitability and Risk Analysis

The profitability of the hybrid model relies on the fact that it can implement state- dependent temporal strategies. Quantitative evidence indicates that the model learned how to deliberately postpone execution in regimes of high volatility, essentially minimizing execution slippage. This was made possible by curriculum learning in training, where environment difficulty was made more complex incrementally—from

a single-agent environment to full stochastic, multi-agent simulations. Risk-adjusted return metrics also confirm the strength of the hybrid system. The Sortino ratio, penalizing downside risk more severely than the Sharpe ratio, was measured at 2.11, highlighting the model’s success in reducing tail risk. The agents learned implicit control regimes in favor of lower exposure during temporary negative price movements but higher-frequency exposure in low-variance stretches. This action closely recapitulates theoretical constructs in stochastic optimal control, in which agents are required to maximize return subject to VaR or CVaR constraints.

This achievement is important in that it shows that strategies from learning algorithms can compete with hand-coded strategies used in traditional statistical arbitrage or market- making research. The emergent behavior of adaptive throttling and execution scheduling in the hybrid framework further suggests the existence of a potential capability for al

# Bibliography

1. Fischer Black and Myron Scholes. The Pricing of Options and Corporate Liabil- ities. *Journal of Political Economy*, 81:637–654, 1973.
2. Hans Bühler, Lukas Gonon, Josef Teichmann, and Ben Wood. Deep Hedging.

*Quantitative Finance*, 19(8):1271–1291, 2019.

1. Steven Heston. A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options. *The Review of Financial Studies*, 6(2):327–343, 1993.
2. Igor Halperin. QLBS: Q-Learner in the Black-Scholes(-Merton) Worlds. *arXiv preprint*, 2017.
3. James M Hutchinson, Andrew W Lo, and Tomaso Poggio. A Nonparametric Approach to Pricing and Hedging Derivative Securities Via Learning Networks. *The Journal of Finance*, 49:851–889, 1994.
4. Petter Kolm and Ritter Gordon. Dynamic Replication and Hedging: A Reinforce- ment Learning Approach. *The Journal of Financial Data Science*, 1(1):159–171, 2019.
5. Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde- Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adver- sarial Networks. *arXiv preprint*, 2014.
6. Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein Generative Adversarial Networks. *Proceedings of the 34th International Conference on Ma- chine Learning*, PMLR 70:214–223, 2017.
7. Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved Training of Wasserstein GANs. *arXiv preprint*, 2017.
8. Olof Mogren. C-RNN-GAN: Continuous recurrent neural networks with adver- sarial training. *arXiv preprint*, 2016.
9. Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. Time-series Genera- tive Adversarial Networks. *NeurIPS Proceedings*, 2019.
10. Mohammad Diqi, Marselina Endah Hiswati, and Adri Saputra Nur. StockGAN: robust stock price prediction using GAN algorithm. *International Journal of Information Technology*, 14:2309–2315, 2022.
11. Magnus Wiese, Lianjun Bai, Ben Wood, and Hans Buehler. Deep Hedging: Learning to Simulate Equity Option Markets. *arXiv preprint*, 2019.
12. Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W. Kempa-Liehr. Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package). *Neurocomputing*, 307:72–77, 2018.
13. Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE.

*Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

1. Hans Bühler, Baranidharan Mohan, and Ben Wood. Deep Hedging: from Theory to Practice, 2019.
2. Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. *arXiv preprint*, 2013.
3. John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint*, 2017.
4. Jiayi Du, Muyang Jin, Petter Kolm, Gordon Ritter, Yixuan Wang, and Bofei Zhang. Deep Reinforcement Learning for Option Replication and Hedging. *The Journal of Financial Data Science*, 2(4):44–57, 2020.
5. Victor Storchan, Svitlana Vyetrenko, and Tucker Balch. MAS-GAN: Adversarial Calibration of Multi-Agent Market Simulators. 2020.
6. Jay Cao, Jacky Chen, John Hull, and Zissis Poulos. Deep Hedging of Derivatives Using Reinforcement Learning. *The Journal of Financial Data Science*, 3(1):10– 27, 2021.
7. Hans Bühler, Phillip Murray, Mikko S. Pakkanen, and Ben Wood. Deep hedging: learning to remove the drift. *risk.net*, 2022.