Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2025) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

37th European Modeling & Simulation Symposium (EMSS 2025), held within the 22nd International Multidisciplinary Modeling & Simulation Multiconference (I3M 2025)

StockMARL: A Novel Multi-Agent Reinforcement Learning System to Dynamically Improve Trading Strategies

Peiyan Zou, Peer-Olaf Siebers*

School of Computer Science, University of Nottingham, Nottingham, UK

Abstract

Contemporary financial markets, shaped by geopolitical tensions, technological innovation, and economic shocks, require adaptive trading strategies. We propose a novel framework that integrates multi-agent simulation with Reinforcement Learning (RL), allowing RL agents to acquire trading strategies by observing heterogeneous rule-based and reactive agents that emulate real-world investor behaviors such as day trading, momentum chasing, and risk aversion. To evaluate this approach, we introduce StockMARL, a multi-agent RL stock market simulation platform. Experimental results show that our observational diversity learning approach has advantages compared to models trained on historical price data and other MARL methods, with RL agents effectively deriving resilient strategies by learning from diverse reactive agent behaviors.

Keywords: stock markets; portfolio optimization; trading strategies; multi-agent simulation; reinforcement learning

1. Introduction

The instability and complexity of contemporary financial markets, intensified by geopolitical tensions, rapid technological advancements, and systemic economic shocks such as the COVID-19 pandemic, demand innovative trading strategies that leverage cutting-edge technologies. The pandemic not only disrupted global markets but also transformed investor behavior, prompting a surge in retail trading and a diversification of investment strategies ranging from index funds to speculative options. This shift, particularly among younger investors, underscores the necessity for financial models to evolve in response to changing market dynamics, enhancing both forecasting accuracy and portfolio optimization.

Reinforcement Learning (RL) and Multi-Agent Simulation (MAS) have both emerged as powerful methodologies in financial modeling; however, existing studies typically focus on training Deep RL models [1, 2, 3] or employ single-

* Corresponding author.
E-mail address: peer-olaf.siebers@nottingham.ac.uk

agent frameworks based primarily on historical market data or simplified simulations [4, 5, 6]. In contrast, Multi-Agent RL (MARL) models have the potential to model dynamic, adaptive interactions between heterogeneous agents in realistic market ecosystems, capturing emergent phenomena like herding, liquidity crises, or strategic collusion that traditional single-agent or static models cannot [7]. While MARL has been explored, prevailing models often involve identical or cooperative agents [8, 9], failing to accurately represent real-world trading environments' competitive and heterogeneous nature. Consequently, these models inadequately capture the complex interactions and behavioral diversity characteristics of actual financial markets.

To address these limitations, we propose a novel stock market simulation system (StockMARL), consisting of an advanced MAS component that incorporates distinct reactive agents, each representing specific financial behaviors reflective of real-world investor types, such as risk-averse investors, trend followers, momentum traders, and day traders. Additionally, a Deep Q-Network (DQN) RL model operates as an individual trader capable of adapting and adjusting its own strategy dynamically within this diverse and reactive simulation context.

2. Background

ML has become a powerful tool in financial modeling, particularly for tasks such as stock price prediction, volatility estimation, and portfolio optimization. Techniques such as Long Short-Term Memory networks, Convolutional Neural Networks, and ensemble tree models have demonstrated efficacy in extracting non-linear patterns from high-frequency market data [1, 10, 11]. However, models trained purely on historical data often suffer from overfitting and poor adaptability to regime shifts or rare market events. Financial markets are dynamic systems influenced not only by price trends but also by behavioral, macroeconomic, and geopolitical factors. Static data-driven approaches therefore struggle to capture emergent, non-stationary dynamics.

To overcome these issues, researchers began exploring RL models, a subset of ML that enables agents to learn policies through trial-and-error interactions with an environment. DQN, Deep Deterministic Policy Gradient, and Proximal Policy Optimization are among the most widely applied RL algorithms in this domain [12]. These models allow the RL agent to make sequential trading decisions in response to changing market states, optimizing cumulative reward such as profit, Sharpe ratio, or risk-adjusted returns. Nonetheless, most RL-based portfolio systems continue to rely on fixed, exogenous market environments, often failing to represent the competitive and adaptive behaviors observed in real financial ecosystems [13]. MAS has emerged as a promising alternative, enabling researchers to replicate market microstructures from the bottom-up. By simulating interactions of heterogeneous agents with different strategies, beliefs, or constraints, MAS captures realistic patterns such as fat tails, clustered volatility, and flash crashes [14, 15, 16]. Despite this progress, many studies still use static or oversimplified agent rules. A persistent challenge is modeling behavioral diversity. Differences in risk appetite, reaction to volatility, and time horizons are essential for reproducing authentic market dynamics. As Lussange et al. argue [9], simulations must model realistic agent heterogeneity and multivariate asset interactions to reflect the feedback loops observed in real markets.

To enhance the MAS approach, Simulation-Enhanced ML (SEML) merges the flexibility of ML with the realism of simulations [17, 18]. In this paradigm, artificial environments populated by agents generate synthetic data or serve as testbeds for learning algorithms. This hybrid method is valuable in finance, where historical data is often limited and unrepresentative of rare or novel conditions. For example, van der Hoog [19] introduced *docking* ML models into MAS, feeding time-series outputs from artificial markets into deep learning algorithms. Maeda et al. [5] demonstrated how training an RL agent within an MAS allows it to learn more robust strategies, particularly under conditions involving transaction costs and market frictions. This research direction is gaining traction as it overcomes the limitations of purely data-driven models and static simulation setups. Embedding intelligent learning agents within dynamic, behaviorally rich environments enables exploration of adaptive, resilient trading strategies. It supports stress testing under rare conditions, improves interpretability through explicit agent interactions, and allows RL models to learn from market microstructures rather than relying solely on macro-level price movements. While SEML offers strong advantages, challenges remain. These include limited agent diversity and behavioral realism, overreliance on price-based learning, and lack of transparency in learning processes.

In our research we address these three challenges with the following strategy. Firstly, we introduce a diverse population of reactive agents, each representing a distinct trading style. This diversity enriches the simulation environment, enabling the RL agent to learn from a wide spectrum of behaviors and improve its adaptability. Secondly, we adopt

a behavior-driven learning paradigm. The RL agent does not rely only on price-based learning but also derives insight from observing the decisions and performances of reactive agents, allowing it to dynamically adapt to market conditions. Finally, we improve transparency by enabling stepwise history tracking within the simulation. This allows evaluation of both reactive and RL agent performance, making simulation dynamics traceable and easier to interpret.

3. Design

To facilitate our exploration, we have designed a heterogeneous MAS platform where diverse reactive trading agents operate under distinct rule-based strategies, generating enriched and realistic trading behaviors within a multi-stock trading environment. We then embedded a DQN-controlled RL agent that derives its decision-making from observing peer agent behaviors and financial indicators.

3.1. System Overview

Our platform is designed with a modular architecture comprising two distinct layers: the Control Layer and the Simulation Layer, as shown in Figure 1. The control layer, referred to as the Artificial Lab, manages episodic training cycles, simulation resets, action encoding and decoding, and serves as the primary interface with the DQN learning algorithm. This layer facilitates seamless communication between the RL model and the underlying simulation engine. The simulation layer, featuring a Stock Market Simulation Model, operates as the internal engine responsible for simulating realistic trading dynamics using the real-world stock data. It manages agent lifecycles, market state progression, and centralized trade logging. Within this simulated environment, a diverse population of reactive agents interacts with a DQN-controlled RL agent under shared market conditions. This integrated design ensures synchronized learning and market simulation, allowing the DQN model to adapt its policy based on agent-driven behavioral signals in a realistic financial environment.

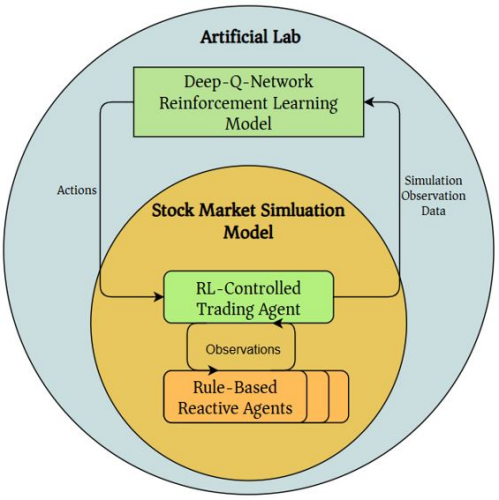

Fig. 1. Platform high level design.

3.2. Reactive Agents

Reactive agents are behaviorally modeled to emulate distinct trader profiles observed in real-world financial markets. Each reactive agent is governed by a rule-based decision-making strategy that reflects its behavioral type and sensitivity to specific market signals. These strategies are carefully engineered to replicate common trader archetypes, thereby contributing to the overall realism and heterogeneity of the simulated market environment. Each agent maintains a persistent internal state, including a live portfolio and trading history.

The *Random Buyer* agent simulates market noise and non-strategic investor behavior to reflect unpredictable market activity often associated with irrational or uninformed participants. At each timestep, it randomly selects an action for every stock with default probabilities. Its actions are constrained by available budget and portfolio holdings to ensure financially valid decisions.

The *Day Trader* agent replicates short-term speculative traders who seek to profit from rapid price movements within a few trading days. It operates with high sensitivity to short-term volatility, momentum shifts, and recent trends, making frequent trading decisions in pursuit of quick gains. Unlike more strategic agents, this type emphasizes speed and reaction over long-term fundamentals.

The *Momentum Trader* agent emulates medium-term technical traders who seek to capitalize on prevailing price trends, adopting the popular "buy high, sell higher" strategy. These agents track trend direction and strength by analyzing the crossover of short-term and long-term moving averages. Unlike short-term day traders, momentum traders take a slightly longer view, entering and exiting positions when sustained directional movement is detected.

The *Risk Trader* agent represents an aggressive, high-risk trading strategy that seeks to exploit extreme price movements and volatile market conditions. Designed to mimic opportunistic and speculative investor behavior, this agent reacts strongly to large price swings, operating with looser decision thresholds, larger position sizes, and a higher tolerance for drawdown and volatility.

The *Risk-Averse Trader* agent represents a conservative investor profile that prioritizes capital preservation and portfolio stability over aggressive returns in direct contrast to the Risk Trader agent. It avoids high-volatile environments, trades infrequently, and operates under strict thresholds for profit-taking and stop-loss, emulating the behavior of institutional funds and risk-sensitive individual investors as a contract to Risk Trader agent.

The *Herding Trader* agent models the behavioral tendency of traders who follow the majority, mimicking observed crowd-driven dynamics in financial markets. Rather than relying on direct price signals or personal portfolio performance, this agent bases its decisions on the collective actions of other agents trading the same stock. Its goal is to align with prevailing market sentiment, if the majority direction signals valuable information.

3.3. Reinforcement Learning Agents

The RL agent uses the DQN algorithm, a value-based method particularly well-suited for environments with discrete and combinatorial action spaces for the multi-stock trading decisions. The observation space is meticulously engineered to provide the RL agent with a comprehensive, behavior-driven snapshot of the simulated financial market environment. It integrates market signals, self-assessment metrics, and peer behavioral dynamics into a high-dimensional observation vector with rich semantic structure. The DQN Model action space is designed to enable the RL agent to make discrete trading decisions across multiple assets, reflecting the real-world constraint of per-stock trade selection within each timestep. For each stock in the simulation, the RL agent may choose one of three possible actions: Buy (the agent places an order to purchase the current stock ticker at current day's open price), Sell (the agent issues a sell order for the current stock ticker at current day's open price), or Hold (the agent refrains from trading for the current stock).

The reward mechanism for the RL agent is carefully engineered to support both short-term trading objectives and long-term portfolio optimization, while encouraging behavioral realism and penalizing impractical trading tendencies. It leverages the RL agent's own performance metrics and relative benchmarking against reactive agents (in observation) to guide the agent's learning trajectory. The reward function is composed of two core components: Performance Reward (short-term and long-term) and Constraint Cost. These elements are combined into a unified scoring mechanism that shapes the RL agent's learning process. Weights are used as scaling parameters that balance competing objectives within the reward function, adjusting the relative importance of performance indicators to guide the RL agent's priorities. The reward function is evaluated at each decision step, providing immediate feedback through short-term rewards and cost, while incorporating cumulative measures that capture long-term portfolio performance across episodes.

The short-term reward component captures the RL agent's immediate trading performance, combining three key indicators: Win Rate, Profit/Loss (P/L), and Daily Money-Weighted Rate of Return (MWRR).

$$Reward_{short} = w_1 \cdot WinRate + w_2 \cdot PL + w_3 \cdot MWRR_{daily} \tag{1}$$

The long-term reward component evaluates the agent's overall investment quality, consistency, and risk management. It integrates the Performance Score (a composite of profitability, cumulative return, and stability), the Concentration Penalty (a deduction for overly concentrated portfolios), and Trade Volatility (a measure of instability in trading behavior).

$$Reward_{long} = w_4 \cdot CumulativeReturn + w_5 \cdot YearlyMWRR + w_6 \cdot PerformanceScore \\ -w_7 \cdot ConcentrationPenalty - w_8 \cdot TradeVolatility \tag{2}$$

The constraint cost component penalizes the RL agent for financially infeasible or behaviorally undesirable trading decisions by combining the accumulated Invalid Action Penalty at the current step (such as buying without budget, selling without holdings, or excessive holdings) with an additional Liquidity Penalty that is applied when the agent maintains a large portfolio but insufficient cash.

$$ConstraintCost = w_9 \cdot \sum_{t=1}^{T} InvalidActionPenalty_t + w_{10} \cdot LiquidityPenalty \tag{3}$$

For the weighting scheme we defined the following principles: The short-term reward weights should help place greater emphasis on Win Rate to encourage accurate anticipation of market trends. The long-term reward weights should help to prioritize cumulative return as the main driver of long-term growth, with balanced contributions from Yearly MWRR and performance score, while applying lighter penalties to concentration and volatility to discourage excessive risk-taking. The constraint cost weights are chosen to balance invalid trade actions against liquidity risks, ensuring the RL agent avoids unrealistic or unstable trading behavior. Currently, these are not in use. For our reward function we used the following weights based on trial and error: $w1 = 0.4$; $w2 = 0.3$; $w3 = 0.3$; $w4 = 0.4$; $w5 = 0.2$; $w6 = 0.2$; $w7 = 0.1$; $w8 = 0.1$; $w9 = 1$; $w10 = 1$.

To stabilize the learning, the reward function is normalized before being passed to the DQN. Unbounded metrics, including all financial metrics, are scaled using the hyperbolic tangent function, which constrains extreme values within $[-1, 1]$ and prevents destabilizing Q-value spikes while preserving relative performance differences. Bounded metrics (including all rates) are instead linearly scaled, as they are naturally bounded between 0 and 100 percent and less prone to extreme outliers, allowing finer sensitivity within their range.

Finally, the reward function passed to the DQN is

$$Reward_{total} = Reward_{short} + Reward_{long} - ConstraintCost \tag{4}$$

3.4. Simulation Process

The simulation process is structured into distinct phases that coordinate interaction between the MAS stock market environment and the RL agent. It begins with system-wide initialization, followed by the episodic training loop. Each episode represents a complete trading lifecycle composed of discrete timesteps corresponding to trading days. Within each timestep, the DQN model receives the current observation state, predicts an action, and the environment returns the decoded action to the market model. Next, the stock market model updates prices, executes trades, and computes new states based on all agent actions. After that, each agent makes trading decisions according to its strategy, contributing to market dynamics, while the stock model logs the trade history. Then, the environment calculates rewards and financial indicators, which are returned to the RL model for learning and policy updates. At the end of each episode, the simulation stores trading histories, performance metrics, and model states. The RL model may update or save parameters depending on the training stage.

3.5. Experiment Design

To define experiment scenarios we identified the following experimental factors to be relevant: number of distinct agent categories (strategy types); number of reactive agents per type (controls group/population size); number of RL agents (adaptive participants); total number of simulation steps (lifecycles); length of each episode in discrete timesteps (trading days); Random number stream seed used for experiment reproducibility; number of distinct stocks included in the simulation; initial budget per agent at the start of each lifecycle. To evaluate the output, we have identified the following responses to be relevant: recorded actions (buy, hold, sell) by agent ID; ticker/stock price today when each agent executes a trading action; individual portfolio data for each agent; financial metrics such as P/L, MWRR, Win Rate, and Performance Score for each agent.

4. Implementation

For the evaluation of StockMARL, we implemented a prototype platform in Python. The structure of this platform is presented in Figure 2. The simulation layer, containing the stock market model and its components (abstract agent base class and concrete agent sub classes with trader profile definitions) is implemented using AgentPy [20], a Python-based framework that supports the creation of heterogeneous populations of reactive trading agents. Each agent operates according to a distinct rule-based strategy, transforming raw market data into simplified, behavior-driven trading signals. These agents collectively shape asset-level trends and market liquidity through their daily trading actions (buy; sell; hold), forming a dynamic simulated environment.

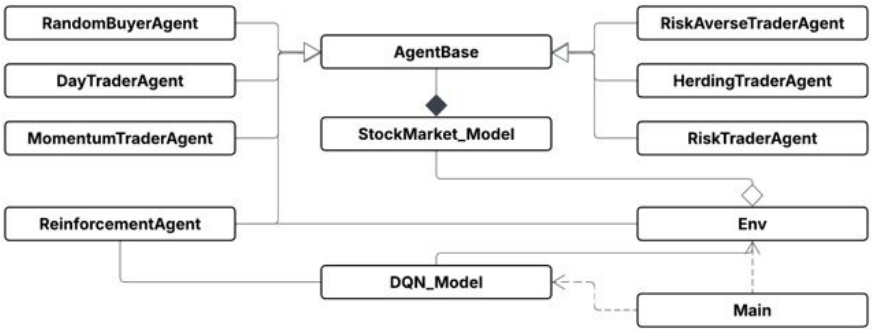

Fig. 2. Platform class diagram.

To support the RL component of the control layer, a custom Gymnasium-compatible environment is employed. Gymnasium [21], a standard Python interface for RL environments, is used to wrap the AgentPy-based simulation, enabling episodic training and seamless interaction between the simulation and the RL agent. Within this setup, the system leverages Stable-Baselines3 [22], a widely used library that provides implementations of cutting-edge RL algorithms. The RL agent is trained using a DQN with a Multi-Layer Perceptron policy, allowing it to operate in the same simulated market as the reactive agents and learn effective trading strategies through behavioral observation. The RL agent's observation space includes both market-level indicators and peer agent behaviors, while the reward function is designed to encourage not only short-term trading performance but also long-term financial goals such as sustained profitability and capital growth. This implementation delivers a scalable and extensible MARL framework capable of exploring how behavioral signals and inter-agent dynamics can inform adaptive, learning-based trading strategies. It also supports in-depth experimentation with diverse trading behaviors under realistic multi-stock trading conditions.

5. Training

To simulate a realistic and diversified financial market environment, we utilize a curated subset of S&P 500 stocks [23]: specifically, AAPL (Technology), META (Communication Services), XOM (Energy), and VISA (Financials). These stocks have been chosen to reflect a broad spectrum of industry sectors and market behaviors. Historical price data was retrieved from Yahoo Finance [24], spanning a decade of market activity up to 1 December 2024, capturing contemporary trends and volatility in the stock market. The dataset is partitioned into training (70%), validation (15%), and testing (15%) subsets, ensuring temporal consistency and enabling robust model evaluation. This separation helps ensure that the model is trained on one portion of the data, fine-tuned on another, and finally tested on unseen data to fairly assess its real-world performance.

6. Experimentation

The evaluation focuses on the following research questions to systematically assess trading performance and strategic adaptability of the proposed system: **Experiment Question 1**: Can the proposed MARL framework improve the

interpretability and efficiency of RL trading strategies compared to traditional RL model training? **Experiment Question 2**: Can the proposed MARL framework effectively adapt to varying market conditions and optimize portfolio management across multiple assets and investment horizons?

To address these questions, the evaluation centers on two main experimental factors: agent population diversity and lifecycle configuration, to assess their impact on system behavior and RL performance. Due to the computational cost of extensive parameter exploration, other variables were predefined in the prototype design. For example, each agent's initial budget was set to $100,000 to ensure a consistent starting point. System effectiveness was measured using key response variables, including agent actions and portfolio metrics such as profitability score and cumulative return, all logged in the simulation history for analysis.

To validate the simulation environment and determine an effective reactive agent mix, a two-stage evaluation process was implemented. In the *validation phase*, population diversity experiments were run, each involving 20 simulation episodes of 252 trading timesteps, totaling 5,040 simulation days. These shorter runs explored the impact of different agent combinations on market dynamics and RL learning, while avoiding the long runtimes seen during prototyping. Earlier experiments showed that increasing total steps significantly extended runtime on current hardware, making shorter runs a necessary trade-off.

Building on the insights gained from the validation phase, a more stable and representative agent mix was formulated and applied in the subsequent *performance evaluation phase*, involving a long-run training experiment spanning 700×252 timesteps. This extended training period, equivalent to approximately 176,400 trading days, enabled a more thorough assessment of the RL agent's learning ability, market adaptability, generalisation capacity, and resilience to overfitting, particularly given the limitations of the training dataset, which covers only seven years of historical stock data. This dual-phase approach ensured that both parameter tuning and final performance analysis were conducted under methodologically rigorous and computationally efficient conditions.

6.1. System Validation through Population Diversity Testing

In our first experiment, StockMARL was evaluated using the validation dataset under various population configurations to identify an optimal balance between agent diversity and computational efficiency. Multiple test scenarios were explored, each with different proportions of reactive agents, ranging from uniform distributions to strategically weighted setups. The tested scenarios were defined by a chain of numbers between 0 and 10, representing the number of reactive agents for each type in the following order: Random Buyer agents, Day Trader agents, Momentum Trader agents, Risk Trader agents, Risk-Averse Trader agents, and Herding Trader agents. The scenarios are as follows:

- 1 1 1 1 1 1 (Baseline Minimal): A single agent from each reactive agent type, providing a basic setup to validate system functionality and isolate individual agent behaviors.
- 3 3 3 3 3 3 (Moderate Diversity): Three agents per agent type, introducing variability within each group through stochastic design, allowing exploration of intra-type behavior.
- 10 10 10 10 10 10 (High Diversity): 10 agents per agent type, introducing variability within each group through stochastic design, allowing exploration of intra-type behavior.
- 1 4 2 3 4 2 (Strategy-Biased): Skewed agent distribution favoring Day Trader agents and Risk-Averse Trader agents, while reducing Random Buyer agents and Herding Trader agents. This aims to explore how increased strategic and conservative behaviors impact market balance.
- 5 3 1 3 1 1 (Uncertainty-Driven): A higher proportion of Random Buyer agents and moderate emphasis on Day Trader agents and Risk Trader agents. This aims to induce greater behavioral uncertainty and short-term speculation in the environment and test the RL agent's adaptability under noisy and opportunistic market conditions.

The experiments revealed that certain agent types, such as Risk Averse Trader agents and Day Trader agents, played a more significant role in guiding the RL agent's learning, while excessive populations of similar agents led to redundant behavior patterns and computational overhead. We can see notable differences across the tested configurations. Table 1 summarizes the performance of the DQN-controlled agent under each of the five population diversity scenarios. Each is reported in the format mean ± std, this representation helps assess not only the average performance of the agent across episodes. Given that each episode simulates distinct market conditions at different time periods, this format provides valuable insights into the agent's consistency and resilience across varying scenarios.

Table 1. DQN-controlled agent summary

Population Param	1 1 1 1 1 1	3 3 3 3 3 3	10 10 10 10 10 10	1 4 2 3 4 2	5 3 1 3 1 1
Portfolio Value	24248.6±13479.8	53159.1±22411.1	32307.5±14589.6	52684.8±20175.6	70567.5±33717.8
Net Worth	93871.8±2593.6	99229.9±4941.1	107997.7±6344.0	92595.6±8014.9	102410.5±4857.9
Daily MWRR (%)	-6.2±2.5	-0.9±4.1	8.0±6.5	-7.5±8.0	2.2±4.9
Monthly MWRR (%)	-6.4±1.9	-1.5±4.4	7.6±7.4	-7.9±8.1	1.4±5.5
Quarter MWRR (%)	-6.9±2.2	-2.5±4.5	6.8±8.2	-8.9±8.3	-0.4±6.2
Yearly MWRR (%)	-9.2±10.4	-4.5±5.6	4.9±7.8	-14.0±11.0	-7.9±5.2
Cum. Return (%)	-6.1±2.6	-0.8±4.9	8.0±6.3	-7.4±8.0	2.4±4.9
Realised Profit/Loss	-7581.3±3619.6	104.0±1334.3	3922.6±3992.7	-2000.2±976.7	-146.0±112.5
Unrealised Profit/Loss	1453.2±2774.9	-874.1±3934.9	4075.1±3173.3	-5404.2±7764.4	2556.5±4892.2
Win Rate (%)	10.0±3.9	18.4±4.1	22.5±9.1	7.8±3.4	10.4±5.2
Profitability Score	-4.3±1.5	-1.9±4.3	4.4±1.9	-4.4±1.5	-2.9±3.4
Trade Volatility	0.2±0.1	0.2±0.1	0.2±0.0	0.2±0.1	0.1±0.1
Performance Score	-4.0±1.5	-1.2±3.4	4.6±2.6	-4.4±2.8	-0.8±2.0

Among the setups shown in Table 1, ”10 10 10 10 10 10”, where each collection of reactive agent types contained 10 members, provided the most balanced and robust performance of the RL agent. It achieved the highest Cumulative Return (8.0%), along with strong results in Yearly MWRR (4.9%) and a high Performance Score (4.6). Meanwhile, the ”5 3 1 3 1 1” scenario offered the second-best performance in general while favoring strategic agent types, whereas the ”1 1 1 1 1 1” baseline scenario resulted in the weakest overall performance, likely due to limited exposure to behavioral variation.

6.2. Generalization through Weighted Population Design

The insights gained from the first experiment directly informed the design of the second experiment. The validation results revealed that population diversity and agent configuration significantly impacted the RL agent’s learning quality, with certain agent types, such as Risk-Averse Trader agents and Day Trader agents, playing a more pivotal role in guiding the agent’s strategy development. This observation highlighted the importance of maintaining a balanced mix of reactive agents, avoiding excessive repetition of similar behaviors, which led to redundant learning signals and computational overhead.

In response to these findings, the final experimental setup employed a strategically weighted agent population, including 5 Random Buyer agents, 7 Day Trader agents, 6 Momentum Trader agents, 6 Risk Trader agents, 7 Risk-Averse Trader agents, 3 Herding Trader agents, and 1 RL agent. This refined configuration maintains a diverse range of strategic behaviors, providing the RL agent with a rich and varied learning environment while ensuring computational feasibility. The emphasis on a higher count of Risk Averse Trader agents and Day Trader agents further enhances the model’s robustness by exposing the RL agent to a mix of cautious and opportunistic trading strategies.

The evaluation was conducted using a dedicated testing dataset, with the simulation results summarized in Table 2. The RL agent demonstrated a consistently strong and balanced performance across various metrics. Notably, it achieved the highest Yearly MWRR (12.2%), while maintaining positive MWRR values across all other periods. Although its Cumulative Return (9.3%) was not the highest among the agents, it remained highly competitive and exhibited a lower standard deviation, indicating more stable long-term performance. However, its Performance Score (5.2) was only average, but had the lowest standard deviation.

Crucially, the RL agent achieved the highest Profitability Score (4.9), which was not only the highest value recorded but also showed the lowest variance among all agents. This result highlights the agent’s consistent profit generation and effective risk management. The RL agent also demonstrated disciplined and stable trading behavior, reflected in a low Trade Volatility of 0.2, among the best across all evaluated agents. In terms of overall performance, the RL agent consistently ranked among the top three, with the lowest standard deviation, further underscoring its reliable and repeatable strategic execution.

Table 2. Testing dataset performance summary (excluding Herding Trader agent and Momentum Trader agent, due to space constraints)

Agent Type	RL	Day Trader	Random Buyer	Risk-Averse Trader	Risk Trader
Portfolio Value	40906.1±15464.9	35920.5±17177.9	83417.4±27502.4	91567.5±26905.4	70715.6±53611.2
Net Worth	109320.2±6008.7	105459.6±5772.9	114736.5±12859.1	108049.3±10498.3	116821.7±19002.1
Daily MWRR (%)	9.4±6.3	5.4±7.0	14.4±24.9	8.0±11.4	16.8±19.2
Monthly MWRR (%)	9.8±7.4	2.9±10.2	10.3±29.3	8.0±14.5	16.2±20.5
Quarter MWRR (%)	10.9±9.6	-3.3±12.1	4.7±31.8	8.4±18.0	14.6±23.3
Yearly MWRR (%)	12.2±18.9	-22.0±12.2	-15.4±39.4	4.5±28.9	5.8±30.1
Cum. Return (%)	9.3±6.0	5.5±5.8	14.7±12.9	8.1±10.5	16.8±19.0
Realised P/L	6796.2±4674.3	5135.6±5576.5	14496.3±12757.4	4474.5±8737.6	13532.3±16478.1
Unrealised P/L	2524.0±1896.9	324.0±732.2	240.2±1318.0	3574.8±4437.4	3289.4±6089.9
Win Rate (%)	27.0±5.7	59.1±9.1	32.6±4.9	12.5±7.3	30.9±15.8
Profitability Score	4.9±1.6	4.1±2.8	4.6±2.0	1.2±4.5	3.6±3.2
Trade Volatility	0.2±0.1	0.5±0.2	0.5±0.1	0.3±0.2	0.1±0.1
Performance Score	5.2±2.2	3.6±2.7	6.6±4.3	2.9±4.9	6.8±6.6

7. Discussion and Conclusions

In this research, we introduced StockMARL, a novel MARL system, where a DQN-based RL agent learns within a dynamic, behaviorally rich simulation environment populated by diverse rule-based trader agents. This design allows the RL agent to develop adaptive trading strategies by observing the decisions and outcomes of heterogeneous agents, including Day Traders, Momentum Traders, and others. These agents simulate a realistic market environment, providing the RL agent with behavior-driven signals that enhance its decision-making. Unlike conventional ML or RL methods trained on historical price data, StockMARL leverages behavior-based observation. The RL agent receives a structured observation space incorporating both market signals and peer behaviors. This enriched input enables it to learn not just from price fluctuations but also from the successes and failures of other traders, supporting more adaptive and interpretable strategies. Furthermore, StockMARL adopts a dual-layer architecture where the DQN model operates externally as a neural learner, while its decisions are executed by an instantiated RL agent within the AgentPy simulation. This ensures that the agent not only learns from behavioral data but also directly experiences the outcomes of its actions, maintaining coherence and traceability between decisions and impacts.

From the final experiments, the RL agent achieved a Yearly MWRR of 12.2%, a Trade Volatility of 0.2 ± 0.1, a Profitability Score of 4.9 ± 1.6, and a Cumulative Return of 15.9%. However, it could not outperform the Risk Trader agent, which recorded the highest Yearly MWRR (16.3%) and Cumulative Return (16.8%), albeit with significantly higher volatility and inconsistent performance.

In comparison to a traditional RL approach [1], where cumulative returns grew by 10%, StockMARL not only matched those but also demonstrated superior overall performance on similar metrics, highlighting its improved interpretability and operational efficiency. The MAS framework reduced invalid actions and improved trade timing and portfolio rebalancing, echoing recent findings from Yao et al. [25]. This answers **Experiment Question 1**, showing improved interpretability and efficiency. Similarly, our results align with systems like FinVision [7], which also leverage MAS to achieve high returns and Sharpe ratios. FinVision employs LLMs instead of Deep RL and relies on limited training data to reduce complexity. In future work, we also plan to integrate LLMs for MARL, but with emphasis on enhancing reactive agent behavior. LLMs offer advanced contextual reasoning, enabling agents to generate realistic dialogues, make decisions, and adapt dynamically to new scenarios without explicit programming. These capabilities could significantly improve StockMARL's adaptability, decision-making, and realism in multi-agent interactions.

A key strength of StockMARL lies in its interpretability, derived from exposure to diverse behaviors. By observing agents such as Day Traders and Momentum Traders, the RL agent developed a generalized policy effective across assets and regimes. Behavior-driven learning allowed it to detect trends, manage volatility, and avoid reactionary trading. Evaluation runs showed fewer invalid actions and strategically timed trades, echoing Yao et al. [25]. Overall,

StockMARL improved interpretability and adaptability, directly answering **Experiment Question 2** by demonstrating portfolio optimization across assets and conditions.

The current system is a prototype, not yet optimized for large-scale deployment. As complexity grows, the action space expands exponentially, posing challenges for RL. Caution is required when making claims such as time advantages, which must be substantiated by further testing. Several enhancements could improve StockMARL's depth and scalability. These include expanding reactive agent types, integrating event-driven dynamics, enhancing agent logic with formalized strategies, and exploring multi-agent RL architectures. Improvements in policy learning, reward shaping, and input vector design could further enhance adaptability. Additionally, broader datasets and upgraded infrastructure would support larger-scale simulations and more complex interactions.

Source Code

The StockMARL platform code is available from GitHub: https://github.com/peiyan03/Stock-MARL

References

[1] Kabbani T, Duman E. Deep reinforcement learning approach for trading automation in the stock market. *IEEE Access*. 2022;10:93564-74.

[2] Hu YJ, Lin SJ. Deep reinforcement learning for optimizing finance portfolio management. In: *2019 Amity International Conference on Artificial Intelligence (AICAI)*; 2019 Feb; Dubai, United Arab Emirates. Piscataway (NJ): IEEE; 2019. p. 14-20.

[3] Pendharkar PC, Cusatis P. Trading financial indices with reinforcement learning agents. *Expert Syst Appl*. 2018;103:1-13.

[4] Dyer J, Quera-Bofarull A, Chopra A, Farmer JD, Calinescu A, Wooldridge M. Gradient-assisted calibration for financial agent-based models. In: *Proceedings of the 4th ACM International Conference on AI in Finance*; 2023 Nov; Brooklyn, NY, USA. New York: ACM; 2023. p. 288-96.

[5] Maeda I, DeGraw D, Kitano M, Matsushima H, Sakaji H, Izumi K, et al. Deep reinforcement learning in agent-based financial market simulation. *J Risk Financ Manag*. 2020;13(4):71.

[6] Liang Z, Chen H, Zhu J, Jiang K, Li Y. Adversarial deep reinforcement learning in portfolio management [preprint]. *arXiv:1808.09940*. 2018 [cited 2025 May 14]. Available from: https://arxiv.org/abs/1808.09940

[7] Fatemi S, Hu Y. FinVision: A multi-agent framework for stock market prediction. In: *Proceedings of the 5th ACM International Conference on AI in Finance*; 2024 Nov; Brooklyn, NY, USA. New York (NY): ACM; 2024. p. 582-90.

[8] Lussange J, Lazarevich I, Bourgeois-Gironde S, Palminteri S, Gutkin B. Stock market microstructure inference via multi-agent reinforcement learning [preprint]. *arXiv:1909.07748*. 2019 [cited 2025 May 14]. Available from: https://arxiv.org/abs/1909.07748

[9] Lussange J, Lazarevich I, Bourgeois-Gironde S, Palminteri S, Gutkin B. Modelling stock markets by multi-agent reinforcement learning. *Comput Econ*. 2021;57(1):113-47.

[10] Awad AL, Elkaffas SM, Fakhr MW. Stock market prediction using deep reinforcement learning. *Appl Syst Innov*. 2023;6(6):106.

[11] Filos A. Reinforcement learning for portfolio management [preprint]. *arXiv:1909.09571*. 2019 [cited 2025 May 14]. Available from: https://arxiv.org/abs/1909.09571

[12] Rajasekhar N, Radhakrishnan TK, Samsudeen N. Exploring reinforcement learning in process control: a comprehensive survey. *Int J Syst Sci*. 2025;1-30.

[13] Raffin A, Hill A, Gleave A, Kanervisto A, Ernestus M, Dormann N. Stable-baselines3: Reliable reinforcement learning implementations. *J Mach Learn Res*. 2021;22(268):1-8.

[14] LeBaron B. Agent-based computational finance: suggested readings and early research. *J Econ Dyn Control*. 2000;24(5-7):679-702.

[15] Chan NT, LeBaron B, Lo AW, Poggio T. *Agent-based models of financial markets: a comparison with experimental markets. MIT Artificial Markets Project. Working paper.* Cambridge (MA): Massachusetts Institute of Technology; 1999.

[16] Lux T. Herd behaviour, bubbles and crashes. *Econ J*. 1995;105(431):881-96.

[17] Hey J, Siebers PO, Nathanail P, Ozcan E, Robinson D. Surrogate optimisation of energy retrofits in domestic building stocks using household carbon valuations. *J Build Perform Simul*. 2023;16(1):16-37.

[18] Do TT, Nguyen DD, Ho XH, Pham HA, Soriano T. A novel simulation-driven data enrichment approach to improve machine learning algorithm performance. In: *International Symposium on Information and Communication Technology*; 2024 Dec; Ho Chi Minh City, Vietnam. Singapore: Springer Nature; 2024. p. 383-97.

[19] van der Hoog S. Deep learning in (and of) agent-based models: a prospectus [preprint]. *arXiv:1706.06302*. 2017 [cited 2025 May 14]. Available from: https://arxiv.org/abs/1706.06302

[20] AgentPy. AgentPy [Internet]; [cited 2025 May 14]. Available from: https://github.com/jofmi/agentpy

[21] Gymnasium. Gymnasium [Internet]; [cited 2025 May 14]. Available from: https://github.com/Farama-Foundation/Gymnasium

[22] Stable Baselines3. Stable Baselines3 [Internet]; [cited 2025 May 14]. Available from: https://github.com/DLR-RM/stable-baselines3

[23] S&P 500. S&P Dow Jones Indices [Internet]; [cited 2025 May 14]. Available from: https://www.spglobal.com/spdji/en/indices/equity/sp-500/

[24] Yahoo Finance. Yahoo! Finance API [Internet]; [cited 2025 May 14]. Available from: https://github.com/ranaroussi/yfinance

[25] Yao Z, Li Z, Thomas M, Florescu I. Reinforcement learning in agent-based market simulation: unveiling realistic stylised facts and behaviour. In: *2024 International Joint Conference on Neural Networks (IJCNN)*; 2024 Jun; Yokohama, Japan. Piscataway (NJ): IEEE; 2024. p. 1-9.