Reinforcement Learning for Portfolio Management

Aditya Chawla (017653246) Brian Zhang (018231876) Shyam Kannan (014132079)

***Abstract*—Portfolio management requires balancing returns with risk while adapting to dynamic market conditions. Traditionally, research in this field involves using a single reinforcement learning (RL) agent and market data such as stock prices, quarterly earnings, and price-to-earnings ratios, to optimize stock portfolio growth. However, in the real world, successful portfolio management consists of several distinct decision-making steps that cannot be reliably solved as a single optimization problem. Instead, this paper proposes a three-tier hierarchical RL architecture that combines multiple algorithms with dual information streams to handle different layers of decision making. The streams being traditional market data and social media sentiment. We train base agents using four different RL algorithms on separate feature sets, aggregate them through specialized meta-agents, and combine both streams via a super-agent that learns when to favor each information source. The hierarchical system substantially outperforms equal-weighted baselines, individual algorithms, and single-asset strategies while reducing downside risk. Results demonstrate that multi-tier aggregation across algorithms and information sources produces superior risk-adjusted returns compared to conventional approaches. Code available on GitHub.**

I. Introduction

Portfolio management is a central problem in quantitative finance. The goal is to decide how to allocate capital across several assets while responding to changing market conditions. Traditional methods rely on modern portfolio theory and other static optimization techniques, which treat the allocation as a single decision that does not adapt over time. These approaches struggle in real markets, where conditions shift quickly and strategies need to evolve with them.

Reinforcement learning offers a different way to think about this problem. Instead of a one-time choice, the task becomes a sequence of decisions where an agent learns how to adjust its portfolio by interacting with the environment. Recent work has shown that hierarchical reinforcement learning can break the trading process into smaller components handled by different agents [1]. Most of these systems rely only on structured market data like prices and trading volume. They often overlook unstructured information from social media, where investors share opinions, expectations, and sentiment.

At the same time, advances in financial natural language processing have shown that language models can pull useful insights from financial text [6]. Despite this progress, sentiment signals are rarely included in multi-agent reinforcement learning systems. When they are used, they typically act as simple supporting features for a single agent rather than being fully integrated into the decision-making structure.

In this work, we introduce a three-layer hierarchical architecture that brings together different algorithms and different types of information. The base layer trains several reinforcement learning algorithms on two separate streams: market data from Yahoo Finance and sentiment data from Reddit. The meta layer learns how to combine the algorithms within each stream. The final super layer learns how to blend the two streams based on which one is more reliable under current market conditions.

This approach differs from prior research in a few key ways. We train multiple algorithms instead of relying on one, which allows the system to take advantage of their different strengths. We also keep market data and sentiment data in separate pipelines until the final stage so each can be processed in a way that fits its structure. Finally, our experiments use real historical data rather than simulations, which makes the results more representative of how the system might behave in practice.

Our evaluation on US equities and commodities shows that this hierarchical design produces stronger risk-adjusted returns than static diversification, individual reinforcement learning algorithms, and methods that rely on only one type of information. It also reduces maximum drawdown during difficult market periods.

II. Related Work

Zhao and Welsch [1] first proposed the Hierarchical Reinforced Trader (HRT), separating stock selection from execution timing through bi-level agents. By decomposing the complexity of portfolio management into smaller sub-tasks, they demonstrated that using a hierarchical approach could improve performance over single-agent approaches in simulated environments.

Kou et al. [2] explored large language models for quantitative investment, showing LLMs can automate strategy discovery by identifying trading patterns and generating investment strategies. Xiao et al. [3] introduced Trading-R1, which combines LLM reasoning with RL for trading decisions, demonstrating that language models provide reasoning capabilities that enhance contextual decision-making.

Furthermore, Koratamaddi et al. [4] used sentiment from Google News and Twitter inside an adaptive Deep Deterministic Policy Gradient (DDPG) model to show that combining sentiment with price data can noticeably improve portfolio performance. Similarly, Nan et al. [5] trained a Deep Q-Network (DQN) with sentiment extracted from news articles after using a knowledge graph to rank relevant news headlines.

Our work extends these approaches by integrating Reddit sentiment with FinBERT [6] alongside market data, comparing four RL algorithms rather than committing to one, training on real historical data from Yahoo Finance [7] rather than simulations, and implementing a three-tier hierarchy (base $\rightarrow$ meta $\rightarrow$ super) that aggregates across both algorithms and information sources.

III. Data

We use two data sources for our 7-asset portfolio consisting of US equities and commodities.

Market Data: Historical daily prices and volumes from Yahoo Finance [7]. Table I lists the assets and their corresponding ticker symbols. The dataset spans January 2022 through November 2025, split into 3-year training period (2022-2024) and 11-month test period (2025).

TABLE I
PORTFOLIO ASSETS AND YAHOO FINANCE TICKERS

Asset	Ticker	Category
S&P 500	^GSPC	US Large Cap Equity
NASDAQ Composite	^IXIC	US Tech Equity
Dow Jones Industrial	^DJI	US Blue Chip Equity
Russell 2000	^RUT	US Small Cap Equity
Gold Futures	GC=F	Precious Metal Commodity
Silver Futures	SI=F	Precious Metal Commodity
Crude Oil WTI	CL=F	Energy Commodity

Sentiment Data: Reddit discussions collected via PRAW [9] from four investment subreddits: r/stocks, r/investing, r/wallstreetbets, and r/StockMarket. Posts are analyzed using FinBERT [6], a BERT model fine-tuned on financial text to extract sentiment polarity and confidence, classifying them as positive, negative, or neutral.

IV. METHODOLOGY

A. Feature Engineering

From the raw market data, we build two separate feature sets that support the dual-track design of our system.

Data-driven features represent traditional financial indicators computed with 30-day rolling windows. For each timestep, we retrieve market price and volume information from Yahoo finance, then calculate measures such as the Sharpe ratio and Calmar ratio for risk-adjusted returns, the Sortino ratio and maximum drawdown for downside risk, annualized volatility, and the pair-wise correlations between all assets. Together, these form a 56-dimensional feature vector made up of 5 metrics per asset across 7 assets along with 21 unique correlation values.

Sentiment-driven features are generated by processing Reddit posts with FinBERT [6] to capture signals related to investor mood and discussion trends. We collect up to 500 recent posts per subreddit and filter them using asset-specific keywords such as "SPY", "gold prices", or "crude oil". The complete list of keywords can be found in table II. For each asset, we compute a sentiment score reflecting the balance of positive and negative posts classified by FinBERT, sentiment volatility indicating how mixed opinions are, a volatility signal that flags periods of unusual uncertainty, article frequency to measure discussion volume, and a confidence score from the model. This produces a 35-dimensional feature vector based on 5 metrics for each of the 7 assets.

These two feature streams allow the system to make use of both quantitative market behavior and qualitative sentiment information from online discussions.

TABLE II
PORTFOLIO ASSETS AND REDDIT KEYWORDS

Asset	Keywords
S&P 500	'S&P 500', 'SPY', 'SP500', 'S&P', 'VOO'
NASDAQ Composite	'NASDAQ', 'QQQ', 'tech stocks'
Dow Jones Industrial	'Dow Jones', 'DIA', 'DJIA'
Russell 2000	'Russell 2000', 'IWM', 'small cap'
Gold Futures	'gold', 'GLD', 'gold prices'
Silver Futures	'silver', 'SLV', 'silver prices'
Crude Oil WTI	'oil', 'crude oil', 'USO', 'WTI'

B. Environment Design and Reward Formulation

We create three custom Gymnasium environments that match the three levels of our architecture. BasePortfolioEnv is used for training individual agents, MetaAgentEnv combines the decisions of the base agents, and SuperAgentEnv integrates both data and sentiment streams to produce the final allocation.

BasePortfolioEnv works directly with historical market data and moves forward one day at a time. The observation size depends on the feature set. Agents trained on market features receive a 56-dimensional vector, while agents trained on sentiment features receive a 35-dimensional vector. The action space is a 7-dimensional vector of continuous weights in the range $[0, 1]$ that must sum to 1. These represent the portfolio allocation across the seven assets.

The reward function is designed to reflect common goals in portfolio management. It is defined as

$$R_t = r_t - 0.5 \cdot \sigma_t - 2.0 \cdot DD_t \tag{1}$$

where r_t is the daily portfolio return, σ_t is the 30-day rolling volatility, and DD_t is the drawdown from the most recent peak. Volatility and drawdown are weighted more heavily than return, which encourages the agent to favor stable and risk-aware decisions.

Portfolio value evolves deterministically: $V_{t+1} = V_t \cdot (1 + \sum_{i=1}^{7} w_i \cdot r_{i,t+1})$ where w_i are allocation weights and $r_{i,t+1}$ are actual historical returns. Episodes run until data exhaustion, providing approximately 200 trading days during testing after accounting for the 30-day feature window.

C. Action Space and Environment Design

We model portfolio allocations using a continuous action space where each action represents a set of portfolio weights across the seven assets. The action is defined as $a_t \in \mathbb{R}^7$, and each component corresponds to the proportion invested in a specific asset. The weights must be non-negative and sum to 1, which matches the standard long-only constraint used in many real portfolio settings: $a_t \in \{w \in [0, 1]^7 : \sum_{i=1}^{7} w_i = 1\}$.

A continuous action space works well for portfolio management because it allows the agent to make precise adjustments. For example, the agent may choose 23.7% in the S&P 500 or 41.2% in gold. Discrete action spaces would force much rougher decisions, which can limit performance when small rebalancing steps matter.

All agents in our system operate within custom Gymnasium environments (BasePortfolioEnv, MetaAgentEnv, and SuperAgentEnv), each of which follows this action space specification. Training is done through Stable-Baselines3 (SB3) [8], which provides reliable implementations of PPO, SAC, DDPG, and TD3. These algorithms are widely used and tested, which helps ensure that our results reflect actual learning behavior rather than differences in implementation.

D. Reproducibility and Experimental Controls

To support reproducible results, we use a fixed random seed (seed = 42) across all experiments. The seed controls sources of randomness such as environment initialization, weight initialization, and action sampling during training.

Some studies evaluate multiple seeds to measure performance variance, but we use a single seed to keep computation manageable. Our system operates on larger feature spaces and performs daily rebalancing, which makes multi-seed training substantially more expensive.

E. Base Agent Training

We train eight base agents in total, covering four algorithms across two feature types. These algorithms offer different approaches to continuous control. Proximal Policy Optimization (PPO) provides stable on-policy learning, Soft Actor-Critic (SAC) encourages exploration through entropy regularization, Deep Deterministic Policy Gradient (DDPG) learns deterministic policies, and Twin Delayed DDPG (TD3) reduces value overestimation during training.

Figure 1 shows the overall architecture. The base agents, shown in blue, process observations according to the feature stream they belong to. The data-driven agents on the left receive 56-dimensional market features, while the sentiment-driven agents on the right receive 35-dimensional features derived from Reddit posts.

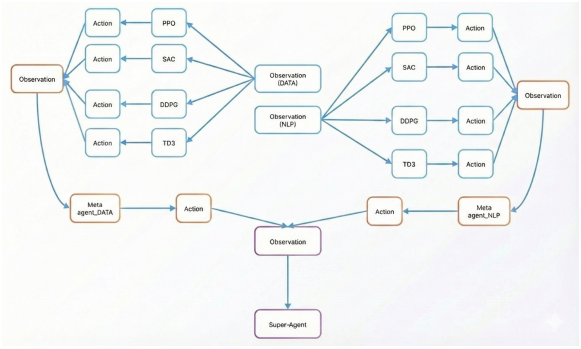

Fig. 1. Three-tier hierarchical RL architecture. Base tier includes eight agents (four data-driven and four NLP-driven) using PPO, SAC, DDPG, and TD3. The meta tier contains two aggregation agents (Meta-Data and Meta-NLP), and the super tier contains a single integration agent that combines both information streams.

Each base agent is trained for 30,000 timesteps using the Stable-Baselines3 [8] implementations. We use the standard hyperparameters provided by the library. PPO uses a learning rate of 3e-4 with 2048-step rollouts, SAC uses a learning rate of 3e-4 with a replay buffer of 100,000 transitions, and both DDPG and TD3 use a learning rate of 1e-3. Training all eight agents takes roughly two hours on standard Google Colab environment.

F. Meta-Agent Aggregation

The second tier of the architecture contains two meta-agents that combine the outputs of the base agents within each information stream. Meta-Agent (Data) receives the seven portfolio weights produced by each of its four data-driven base agents, giving 28 dimensions, along with the current 56-dimensional market feature vector. This results in an 84-dimensional observation. The meta-agent learns how to weight the base agent recommendations based on which algorithms are performing well under current market conditions.

Meta-Agent (NLP) follows the same structure for the sentiment-driven stream. It receives the 28 dimensions from its four base agents and the 35-dimensional NLP feature vector, producing a 63-dimensional observation. Both meta-agents are trained with PPO for 15,000 timesteps and output a seven-dimensional vector of portfolio weights.

This tier acts as a learned ensemble rather than a simple average. By considering both the base agent recommendations and the current feature values, each meta-agent can adjust its weighting strategy and favor the algorithms that are most effective in the market environment.

G. Super-Agent Integration

The super-agent forms the top layer of the hierarchy and brings together the outputs of both meta-agents. Its input is a compact 15-dimensional vector that includes the seven portfolio weights from Meta-Agent (Data), the seven weights from Meta-Agent (NLP), and a single market regime indicator based on recent volatility.

With this limited set of inputs, the super-agent learns broad decision rules. It learns when to rely more on data-driven strategies, when sentiment-driven strategies are more reliable, and how to respond when both sources agree or diverge. It is trained with PPO for 10,000 timesteps and outputs a final set of seven portfolio weights.

The three-tier structure creates a step-by-step refinement process. The base agents explore a range of strategies across different algorithms and feature types. The meta-agents then combine the strategies within each domain. Finally, the super-agent integrates both domains to produce the final allocation.

H. Evaluation Protocol

All agents are evaluated on an out-of-sample test period covering January through November 2025. Each agent begins with the same initial conditions: $100,000 in starting capital, daily rebalancing, and no transaction costs. We report total return, annualized return, Sharpe ratio, volatility, and maximum drawdown to assess both performance and risk.

We compare our agents with two baseline strategies. The first is an equal-weighted allocation that invests one seventh of the portfolio in each asset, representing a simple diversification approach. The second is an S&P 500-only strategy that reflects a concentrated equity position. These baselines allow us to evaluate whether the learned strategies provide meaningful improvements over naive diversification and single-asset exposure.

V. Results

A. Performance Overview

Table III summarizes the performance of all agents and baselines on the test period. The super-agent achieved a total return of 55.07% (74.79% annualized) with a Sharpe ratio of 2.89. This is noticeably stronger than both the equal-weighted baseline, which produced an annualized return of 28.13% with a Sharpe ratio of 1.58, and the S&P 500-only strategy, which returned 18.53% annually with a Sharpe ratio of 0.97.

TABLE III
Performance Comparison Across All Agents and Baselines

Agent	Total ROI (%)	Annual ROI (%)	Sharpe Ratio	Vol. (%)	Max DD (%)
Super	55.07	74.79	2.89	20.03	-9.86
Equal Weighted	25.26	28.13	1.58	16.44	-15.62
DDPG	21.48	28.10	1.55	16.91	-12.14
SAC	18.86	24.59	1.34	17.59	-13.87
TD3	18.18	23.69	1.36	16.72	-13.84
S&P 500 Only	16.71	18.53	0.97	19.32	-18.90
Meta NLP	12.34	15.97	1.01	16.00	-14.81
PPO	6.29	8.07	0.45	23.45	-19.45
Meta Data	2.86	3.65	0.28	19.92	-16.40

Meta-agents act as intermediate aggregation layers rather than full standalone strategies, so their performance should be interpreted as part of the larger hierarchy.

B. Risk-Adjusted Performance

Figure 2 compares Sharpe ratios across all agents. The super-agent achieves a ratio of 2.89, which is almost twice the best-performing base agent (DDPG at 1.55). This highlights the super-agent's ability to generate stronger returns while maintaining lower risk. PPO falls below the commonly used threshold of 1.0, indicating weaker risk-adjusted performance.

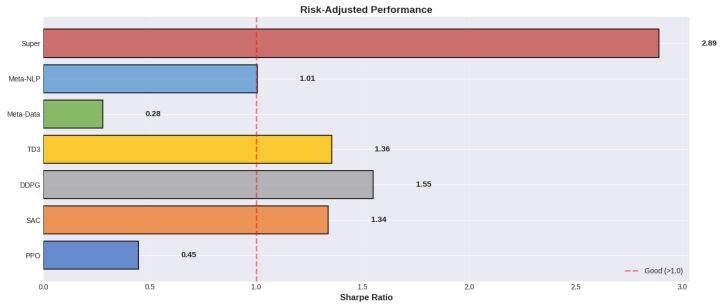

Fig. 2. Sharpe ratio comparison across agents. The super-agent achieves the highest risk-adjusted performance at 2.89. The red dashed line marks the 1.0 threshold often used to indicate acceptable risk-adjusted returns.

The maximum drawdown results also show meaningful downside protection. The super-agent's worst decline is -9.86%, compared with -18.90% for the S&P 500-only strategy and -15.62% for the equal-weighted baseline.

C. Temporal Dynamics

Figure 3 shows cumulative returns over 200 trading days. All agents experienced initial drawdown (days 0-40) before recovering. The super-agent demonstrated strongest recovery and sustained growth, reaching 55% total return while base agents plateaued at 12-21%.

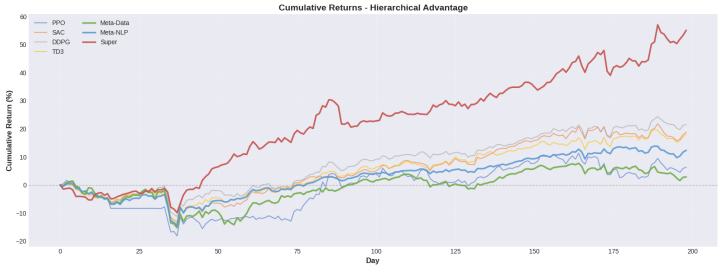

Fig. 3. Cumulative returns over 200-day test period. Super-agent (red) substantially outperforms base agents, meta-agents, and baselines throughout evaluation period.

D. Dynamic Allocation Strategy

Table IV shows how the super-agent adjusts its portfolio in response to changing market conditions. Early in the test period, the agent leans toward equities, allocating 45% to the S&P 500 in February. As volatility increases later in the year, the agent shifts toward commodities, with allocations reaching 69% in gold during October and 42% in oil during June. The willingness to assign 0% to certain assets for long stretches suggests that the agent learns to avoid positions that are not contributing to performance.

TABLE IV
MONTHLY AVERAGE PORTFOLIO ALLOCATIONS (SUPER-AGENT)

Month	S&P 500 (%)	NASDAQ (%)	DOW (%)	Russell (%)	Gold (%)	Silver (%)	Oil (%)
Feb 2025	45.3	0.0	29.4	0.0	12.1	0.0	13.1
Mar 2025	1.3	9.8	13.0	1.9	52.2	8.5	14.2
Apr 2025	29.5	5.3	20.1	2.7	40.1	1.8	2.9
May 2025	24.7	6.1	14.9	0.0	26.8	2.9	24.3
Jun 2025	23.3	11.9	2.7	0.0	24.9	15.3	42.2
Jul 2025	0.3	40.8	0.0	0.0	20.5	18.8	19.6
Aug 2025	1.0	25.6	0.0	0.0	25.6	15.9	30.9
Sep 2025	1.1	32.9	0.0	0.0	49.4	0.0	17.6
Oct 2025	2.4	8.5	0.0	0.0	68.7	7.6	20.8
Nov 2025	18.1	0.0	25.3	6.7	28.7	25.5	1.6

E. Algorithmic Diversity Benefits

Among the base agents, DDPG delivered the strongest performance with an annualized return of 28.10%. SAC and TD3 followed with returns of 24.59% and 23.69%, while PPO performed noticeably worse at 8.07%. The range in performance highlights the value of using multiple algorithms, since no single method performs best in all conditions.

F. Hierarchical Value Addition

The super-agent outperformed the best base agent (DDPG) by a factor of 2.66 in total returns and also reduced maximum drawdown from -12.14% to -9.86%. Each layer of the hierarchy contributes to this improvement. The base agents explore a wide set of strategies, the meta-agents learn to select the most effective approaches within each information stream, and the super-agent brings together the strengths of both streams to produce the final allocation.

VI. CONCLUSION

We developed a three-tier hierarchical reinforcement learning system for portfolio management that combines Reddit sentiment with traditional market data. The system trains eight base agents across two information streams, aggregates them through meta-agents, and produces final allocations through a super-agent. On a portfolio of seven US assets over an eleven-month test period, the super-agent achieved a 74.79% annualized return with a Sharpe ratio of 2.89. This outperformed both the equal-weighted baseline, which returned 28.13% annually with a Sharpe ratio of 1.58, and the best-performing individual base agent, DDPG, at 28.10% annually with a Sharpe ratio of 1.55.

Several observations emerge from our results. The base agents learn noticeably different behaviors, which supports the idea that combining multiple algorithms can improve performance. Sentiment features extracted from Reddit using FinBERT add useful information that is not captured by traditional market indicators. The hierarchical structure strengthens risk-adjusted performance, reducing maximum drawdown from -15.62% in the equal-weighted portfolio to -9.86% for the super-agent. Training on real historical data also helps avoid the optimistic returns often seen in purely simulated environments.

The system extends earlier hierarchical trading frameworks by separating market data and sentiment data into their own learning pipelines. This allows the super-agent to recognize when each source is more reliable and adjust its strategy accordingly.

There are several areas for improvement. The current experiments rely on a small set of US assets and a relatively short training window. The model is also trained using a single random seed, and sentiment is limited to Reddit posts. The reward function uses fixed weights for return, volatility, and drawdown rather than adapting these preferences over time. Expanding the dataset to include global equities, bonds, and cryptocurrencies would allow us to test scalability. Incorporating additional sentiment sources such as Twitter, financial news, and earnings transcripts could further enrich the inputs. Future work may also include transaction cost modeling, multi-seed evaluations with statistical tests, walk-forward validation with periodic retraining, and comparisons with professional fund benchmarks. Attention mechanisms inside the super-agent could improve interpretability.

The architecture is built to be flexible. New algorithms or feature sets can be added at the base layer, new information sources can be paired with additional meta-agents, and the super-agent can scale to combine a wider range of inputs. Overall, our results show that a hierarchical reinforcement learning approach that blends multiple algorithms and information sources can outperform individual agents and simple static strategies. Integrating sentiment with technical indicators improves risk-adjusted returns and offers a promising direction for automated portfolio management.

VII. Individual Contributions

Shyam Kannan (Data Preprocessing and Experimental Setup): Collected and preprocessed financial market data for 7 assets using Yahoo Finance API, including daily closing prices and returns computation. Implemented the Reddit sentiment scraping pipeline using asyncpraw and FinBERT for NLP-driven feature extraction across finance subreddits. Designed and implemented the custom Gym environment (‘BasePortfolioEnv‘) with realistic portfolio constraints (long-only, no leverage, monthly rebalancing). Configured the reward function balancing ROI, volatility, and maximum drawdown with hyperparameter tuning. Managed computational workflows and training pipelines in Google Colab, ensuring proper model check pointing and version control.

Aditya Chawla (Architecture and Training): Designed and implemented the three-tier hierarchical reinforcement learning architecture consisting of base agents, meta-agents, and a super-agent. Trained 8 base agents—4 data-driven (PPO, SAC, DDPG, TD3 on financial metrics) and 4 NLP-driven (PPO, SAC, DDPG, TD3 on sentiment features)—using Stable-Baselines3 with 30,000 timesteps each. Implemented the MetaAgentEnv class in PyTorch to aggregate base agent decisions through custom observation vectors concatenating all base agent weights with market features. Trained two meta-agents (data-driven and NLP-driven) with 15,000 timesteps to specialize on respective modalities. Built the SuperAgentEnv combining both meta-agent outputs with market regime indicators, training the final super-agent with PPO for 10,000 timesteps. Configured all hyperparameters including learning rates, reward weights, and random seed for reproducibility.

Brian Zhang (Testing, Evaluation and Results Analysis): Developed comprehensive evaluation functions to backtest all 11 agents on unseen 2025 test data. Computed performance metrics including total ROI, annualized ROI, Sharpe ratio, annualized volatility, and maximum drawdown for all hierarchical levels. Implemented benchmark comparisons against Equal-Weighted portfolio (1/7 allocation) and S&P 500-only strategies. Created the complete visualization suite including portfolio value evolution curves, monthly allocation table, and comparative performance tables. Analyzed results to identify performance patterns across agent hierarchies and data modalities, documenting strengths and limitations of the hierarchical approach.

Shared Responsibilities (All Members): Collaboratively conducted iterative code review, debugging, and proofreading of each module to ensure integration and reproducibility. Jointly prepared the technical report and presentation slides, with peer verification of experimental results across multiple runs.

References

[1] Z. Zhao and R. E. Welsch, “Hierarchical Reinforced Trader (HRT): A Bi-Level Approach for Optimizing Stock Selection and Execution,” *arXiv preprint*, 2024. [Online]. Available: https://arxiv.org/abs/2410.14927.

[2] Z. Kou, H. Yu, J. Peng, and L. Chen, “Automate Strategy Finding with LLM in Quant Investment,” *arXiv preprint*, 2024. [Online]. Available: https://arxiv.org/abs/2409.06289.

[3] Y. Xiao, E. Sun, T. Chen, F. Wu, D. Luo, and W. Wang, “Trading-R1: Financial Trading with LLM Reasoning via Reinforcement Learning,” *arXiv preprint*, 2025. [Online]. Available: https://arxiv.org/abs/2509.11420.

[4] P. Koratamaddi, K. Wadhwani, M. Gupta, and S. G. Sanjeevi, “Market sentiment-aware deep reinforcement learning approach for stock portfolio allocation,” *Engineering Science and Technology, an International Journal*, vol. 24, no. 3, pp. 848–859, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2215098621000070

[5] A. Nan, A. Perumal, and O. R Zaiane, ”Sentiment and knowledge based algorithmic trading with deep reinforcement learning,” *International Conference on Database and Expert Systems Applications*. Springer, 167–180, 2022. [Online]. Available: https://arxiv.org/pdf/2001.09403

[6] Y. Yang, M. C. S. Uy, and A. Huang, “FinBERT: A Pretrained Language Model for Financial Communications,” *arXiv preprint*, 2020. [Online]. Available: https://arxiv.org/pdf/2006.08097

[7] “Yahoo Finance Dataset (2018–2023),” Kaggle, 2018. [Online]. Available: https://www.kaggle.com/datasets/suruchiarora/yahoo-finance-dataset-2018-2023.

[8] “Stable-Baselines3 Models Collection,” Hugging Face, 2025. [Online]. Available: https://huggingface.co/sb3/models

[9] “Reddit.com: Api Documentation,” www.reddit.com. [Online]. Available: https://www.reddit.com/dev/api/