Stock Portfolio Allocation using LLM News Sentiment and Deep Reinforcement Learning

Triantafillos Papadopoulos
Athens University of Economics and Business
Athens, Greece
trpapadopoulos@aueb.gr

Abstract

In the recent years, large language models and reinforcement learning have each played a pivotal role in driving research forward in their respective areas. Large language models have mainly focused on individual stock trading tasks, with some recent works extending this to portfolio optimization. Financial reinforcement learning on the other hand has been utilized in both individual stock trading and portfolio optimization, often times utilizing external solvers to allocate portfolio weights to each stock in the portfolio. Despite these methods widespread success, their combination remains underexplored. We present our work, LLM-RL-Portfolio-Allocation as a hybrid method, which utilizes LLM trading signals to create features and leverages deep reinforcement learning to allocate the portfolio weights. Our comprehensive evaluation of several deep reinforcement learning algorithms remains challenging but showcases strengths and weaknesses of each algorithm for this specific task. These findings underscore the limitations of a hand-picked stock portfolio, portfolio allocation using deep reinforcement learning and the manual creation of features to be passed in these models.

Keywords

Financial Large Language Models, Deep Reinforcement Learning, Financial Reinforcement Learning, Sentiment Analysis, Portfolio Optimization, Portfolio Allocation, Stock Trading

1 Introduction

Stock trading, be it individual stock trading or portfolio management remains an exciting field of research, with a plethora of publications from multiple domains, such as Finance [11], Natural Language Processing [10, 25] and Statistics [7, 15] . Multi-billion dollar asset management companies and hedge funds, utilize existing research, but also their own private methods to manage their customer's capital, maintaining their wealth, increasing their capital gains while also profiting off commissions for this management. Individual investors also seek to come up with strategies which can beat the market and boost their personal wealth. Therefore, the widespread interest in this area of research and practice is evident.

Despite recent advancements in applying deep reinforcement learning (DRL) to stock trading, the task of portfolio allocation remains largely underexplored. DRL has been extensively used for individual stock trading [4] and portfolio optimization [6, 13] showcasing the use of several algorithms to train deep neural networks adequate for these tasks. Individual stock trading is the task predicting the ideal trading action (Buy, Hold, Sell) given a single stock's information. Portfolio optimization, consists of selecting the best stocks to trade from a wide array of options. Contrary to the previous, portfolio allocation is the task of allocating capital to a specific set of preselected stocks. To the best of our knowledge, most approaches to portfolio allocation, utilize a portfolio optimization step and afterwards allocate the capital to each stock utilizing external solvers, like a mean-variance optimization, instead of utilizing DRL themselves.

On the other hand, Large Language Models (LLMs) have been proven to work particularly well financial tasks, being apple to utilize not just numerical features but textual information. Textual information, particularly news, information from company filings and other sources have been shown to hold strong predictive powers when it comes to predicting stock price movements. Previous work has utilized LLMs to output trading decisions from news articles [18] while more recent approaches utilize a Multi-agent approach [19, 24], using multiple LLMs as agents, each specializing in a particular task and combining their outputs to reach a trading decision.

In this work, we introduce a hybrid approach. First we select a portfolio of 8 handpicked stocks, each from a different sector of the economy. We utilize FinBERT [21], a large language model fine tuned in the task of extracting sentiment from financial news headlines. These sentiment scores, are passed alongside widely adopted technical indicators which showcase stock health and performance, and are passed as features in a deep neural network which outputs the percentage of our total capital to be allocated to each individual stock in the portfolio. This model is then trained using various Deep Reinforcement Learning algorithms and we showcase a comparison of the results for each method.

Our main contributions are as follows:

- We introduce a hybrid approach to portfolio allocation, utilizing LLM signals and Deep Reinforcement Learning.
- We evaluate various deep reinforcement learning algorithms for the task of portfolio allocation.

This implementation is publicly available[1].

2 Related Work

2.1 Financial Reinforcement Learning

In recent years, reinforcement learning has been widely adopted in various financial tasks and particularly stock trading. Liu et al. [9] select 30 stocks as their portfolio. The state consists of the prices of each stock, the amount of holdings for each stock and the remaining balance. The possible actions selected by the policy are Buy/Hold/Sell for each stock in the portfolio. This approach is trained using Deep Deterministic Policy Gradient [8], using the change in total portfolio value as the reward. Xiong et al. [20] propose a mixed method. The state consists of various numerical indicators and historical prices. This state is encoded using an LLM and the encoding is the input to the policy which is trained using

[1] https://github.com/daffypapa/Portfolio-Optimization-using-LLMs-and-RL

Proximal Policy Optimization [16] using the difference in sharpe ratio between two succeeding timesteps as the reward. The LLM also receives gradients and is fine tuned to provide better state encodings. In [1], Benhenda et al. use traditional technical indicators and also use LLMs to generate new features. These features are ordinal scalars valued from 1 to 5, one based on the stock recommendation of the LLM for the particular stock and one based on the LLM's risk assessment of the stock. A policy is then trained for the task of portfolio optimization using Conditional Value-at-risk Proximal Policy Optimization [22], a variant of PPO using the difference in the total portfolio value between two timesteps as the reward.

2.2 Large Language Model Stock Trading

Numerous approaches have utilized large language models in the financial task of stock trading. In [18] Wang et al. use LLMs to extract key factors from news articles combine these with historical stock prices to predict stock movement. Similarly, Fatouros et al. [3] generate a complete investment signal for each stock with the reasoning for the decision, generated by LLM analysis of stock fundamentals, daily news and investment reports. They then use external portfolio allocation techniques based on which stocks received a Short or Long signal based on the LLM analysis. Yu et al. [23] perform single stock trading with an LLM agent approach. Specifically, an LLM analyzes daily market news, company 10-Q, 10-K filings and daily stock prices. These analyses at different timepoints form the agent's working memory. They utilize a layered memory mechanism with self-reflection, where the agent retrieves relevant insights, observes the market movement and reflects upon why the stock price moved the way it did given the analysis collected in the memory module, a process which mimics training. Then, during test time, the agent has access to it's self reflections and uses these to signal an investment decision, comparing it's current analysis with the analyses collected during training. Another approach by Xiao et al. [19] is characterized by it's use of multiple agents. Particularly, their architecture consists of an analyst team, which has an individual LLM agent analyzes inputs from each of the following sources: stock fundamentals, daily news, company filings, technical analysis and social media posts. These collaborate and generate a complete report on a specific stock. Then, a researcher team, which consists of a Bullish and Bearish researcher, receive this report and discuss. The bullish researcher generates reasons to buy, while the bearish researcher generates reasons to sell. They debate for a predefined number of steps or until a debate facilitator agent decides which researcher won the debate. The debate facilitator decides on the trading action and provides reasoning for this decision, based on the researcher team's debate. On top of this module, they also utilize a risk management team and a manager agent which finally authorizes the decision and is responsible for the trade.

3 LLM-RL-Portfolio-Allocation: Our Hybrid Approach

In this section, we introduce LLM-RL-Portfolio-Allocation our proposed hybrid approach.

3.1 Dataset Curation and Preprocessing

The original dataset used in this study is derived from FNSPID [2]. It consists of approximately 15.7 million financial news articles related to 4,775 S&P 500 companies, spanning the period from 1999 to 2023. These articles were aggregated from four prominent stock market news websites. The dataset is notable for its extensive coverage, diversity, and integration of sentiment annotations associated with the news content.

Following a thorough exploratory data analysis, we selected 8 stocks to construct our portfolio. Each stock is a constituent of the S&P 500 and meets two key criteria: a sufficient volume of associated news data and continuous listing on the S&P 500 throughout the 1999–2023 period.

To represent the broader market sectors associated with these portfolio stocks, we additionally selected 3 representative stocks per sector from the same FNSPID dataset. These sector representatives serve as proxies for evaluating sector-level trends and sentiment. The selection ensures sectoral diversity while maintaining data quality and consistency. The stock and sector assignments are illustrated in the figure below.

Sector	Portfolio Stock	Stand-in 1	Stand-in 2	Stand-in 3
Healthcare	ABT	UNH	JNJ	CVS
Consumer Discretionary	AMZN	MCD	F	GM
Financials	BLK	BRK.B	AXP	JPM
Energy	XOM	CVX	DVN	CTRA
Information Technology	NVDA	AAPL	AMD	CSCO
Communication Services	T	CMCSA	TMUS	VZ
Consumer Staples	KO	PG	COST	KMB
Industrials	GE	MMM	DAL	BA

Table 1: Portfolio stocks and their sector stand-ins selected from the FNSPID dataset, ensuring coverage across 8 key S&P 500 sectors.

For each day in the dataset, a company may have multiple news items. In such cases, we concatenate all news summaries related to that company into a single text, which represents the total daily news signal for that company. This aggregated news summary is then processed using FinBERT [21] to extract sentiment scores—positive, neutral, and negative—along with their corresponding confidence levels. These sentiment scores are combined to produce a company-level sentiment feature, where values range from -1 (indicating negative sentiment) to 1 (indicating positive sentiment). To derive sector-level sentiment, we utilize sector stand-ins by averaging the sentiment scores of companies within the same sector, thereby generating a sector sentiment feature for each stock in the portfolio.

Daily stock price data from 2009-01-01 to 2023-12-31 for the portfolio stocks were obtained using Yahoo Finance. The market sentiment and sector sentiment features were then merged with the stock price data, ensuring accurate temporal alignment across all datasets. Finally, commonly used technical indicators, along with the covariance matrix of closing prices for all portfolio stocks, were computed for each trading day in the dataset.

3.2 Method

Considering the stochastic and interactive nature of the trading market, we model the stock trading process as a Markov Decision Process (MDP) which is specified as follows:

- **State** s: a collection of features representing the current market and portfolio conditions. For each of the 8 stocks, the state includes technical indicators, the latest closing price, and company-specific sentiment. Additionally, the state incorporates an overall sector sentiment indicator. These features collectively form the input used to inform portfolio allocation decisions.
- **Action** a: a portfolio allocation vector over all D assets, where each element represents the proportion of total capital assigned to a specific asset. The weights are continuous values between 0 and 1 and sum to 1, reflecting how capital is distributed across the portfolio at a given time.
- **Reward** $r(s, a, s')$: the change in the portfolio performance when action a is taken at state s and the system transitions to state s'. Specifically, the reward is defined as the difference in Sharpe ratios between time t and $t-1$, reflecting risk-adjusted returns. The portfolio value is based on how capital is allocated across assets, where a percentage of the total capital is assigned to each asset according to the selected weights and multiplied by its corresponding price.
- **Policy** $\pi(s)$: the portfolio allocation strategy at state s. It represents a probability distribution over portfolio weights a at state s, where the weights specify the proportion of capital allocated to each asset and sum to 1.

3.3 Training Details

The training dataset included daily data from January 1, 2009, to December 31, 2021. The validation set covered the entire year of 2022, while the backtest was conducted using data from the full year of 2023. The neural network, consisting of 2 fully connected layers of 64 nodes and an output layer of 8 nodes, was trained using four deep reinforcement learning algorithms: A2C [12], DDPG [17], PPO [16], and SAC [5], each for 204,800 timesteps. A range of hyperparameters—such as the number of steps per episode, batch size, learning rate, and other algorithm-specific settings—were tested for each method. Due to computational constraints, we conducted manual tuning by evaluating a limited number of configurations for each algorithm and selecting the one that achieved the best performance on the validation set. The optimal hyperparameter configuration for each algorithm was selected based on the highest Cumulative Return and Sharpe Ratio on the validation set. After selection, the training and validation datasets were combined, and each algorithm was retrained using its optimal hyperparameters for 409,600 timesteps. The best hyperparameter settings for each algorithm are detailed in the appendix.

4 Evaluation

We use four widely recognized financial metrics to evaluate and compare the portfolio allocation performance of the different models trained by the various RL algorithms: Cumulative Return (CR), Sharpe Ratio (SR), Annualized Volatility (AV), and Maximum Drawdown (MDD). Cumulative Return and Sharpe Ratio in particular focus more on long-term gains and risk-adjusted returns and hence are typically considered more important than AV and MDD for evaluation the value of a portfolio and the trading performance. Accordingly, we treat Cumulative Return and Sharpe Ratio as our primary metrics for the final evaluation.

4.1 Cumulative Return

Cumulative Return (CR) % measures the total percentage change in the value of a portfolio over time. When using continuously compounded (logarithmic) returns, it is calculated by summing the natural logarithms of the ratios of consecutive portfolio values:

$$\text{Cumulative Log Return} = \sum_{t=1}^{T} \ln\left(\frac{P_t}{P_{t-1}}\right)$$

Where P_t is the value of the portfolio at day t. To express the total cumulative return as a percentage gain, we exponentiate the sum:

$$\textbf{CR} = \exp\left(\sum_{t=1}^{T} \ln\left(\frac{P_t}{P_{t-1}}\right)\right) - 1$$

4.2 Sharpe Ratio

The **Sharpe Ratio (SR)** measures the risk-adjusted performance of a portfolio. It is defined as:

$$\textbf{SR} = \frac{R_p - r_f}{\sigma_p}$$

where:

- R_p is the average return (the average of daily PnL),
- r_f is the risk-free rate (commonly set to 0 in practice for simplicity),
- σ_p is the standard deviation (volatility) of daily returns.

Here the daily Profits and Losses are simply the difference between portfolio values at day t and t - 1.
Specifically, $PnL_t = P_t - P_{t-1}$ where P_t is the portfolio value at trading day t.

To annualize the Sharpe Ratio, we multiply by $\sqrt{252}$ since there are approximately 252 trading days in a year:

$$\textbf{SR}_{\text{annual}} = \frac{R_p - r_f}{\sigma_p} \times \sqrt{252}$$

4.3 Max Drawdown

Maximum Drawdown (MDD) is a risk metric that quantifies the most significant decline in account balance from a historical peak to a subsequent trough. It is expressed as a percentage and is defined by the following formula:

$$\textbf{MDD} = \max\left(\frac{V_{\text{peak}} - V_{\text{trough}}}{V_{\text{peak}}}\right)$$

A smaller MDD value suggests reduced risk exposure and indicates a more resilient trading or investment strategy.

4.4 Annualized Volatility

Daily Volatility (DV) % and Annualized Volatility (AV) % are used to measure the variability in asset returns. AV is calculated by multiplying DV, which represents the standard deviation of daily log returns, by the square root of the typical number of trading days in a year (assuming 252 trading days in a year):

$$\textbf{AV} = \textbf{DV} \times \sqrt{252}$$

5 Results

In this section, we present the final trading results from the backtesting period covering the full year of 2023. We evaluate the performance of four reinforcement learning (RL) algorithms against three benchmarks: (1) an equally weighted portfolio of all eight stocks, (2) a portfolio with randomly assigned weights to each stock, and (3) the S&P 500, a widely followed index representing the top 500 publicly traded companies in the U.S. stock market. The S&P 500 is often used as a benchmark for overall market performance, and consistently outperforming it is generally considered a strong indicator of successful investment strategies.

Method	CR↑	SR↑	AV↓	MDD↓
Equal Weights	41.478	2.428	**14.420**	8.672
Random Weights	43.999	2.451	14.960	9.908
S&P500	26.709	1.872	13.055	9.974
A2C	45.831	2.337	16.262	8.864
DDPG	45.372	**2.585**	14.720	**7.351**
PPO	28.176	1.703	14.781	8.521
SAC	**49.144**	2.546	15.783	10.360

Table 2: Performance of different portfolio allocation methods on the backtest trading period of 2023.

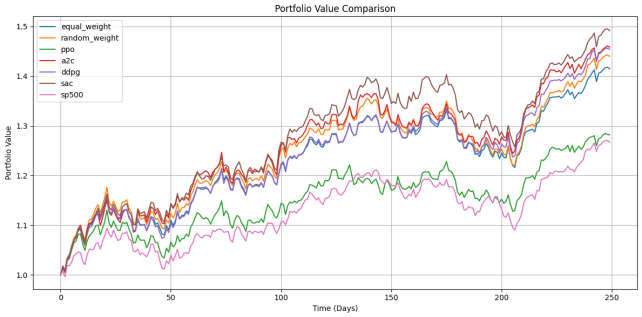

Figure 1: Comparison of portfolio value trajectories over the backtest period for different reinforcement learning algorithms and baseline strategies. All portfolios start with $1,000,000 and are evaluated daily for the entire year of 2023. Note that the returns are normalized in this plot so these returns are irrelevant of the original capital.

We observe that all RL-based strategies outperform the S&P 500 in terms of cumulative return (CR), with SAC achieving the highest return of 49.144%. Notably, DDPG achieves the highest Sharpe Ratio (SR) of 2.585, indicating the best risk-adjusted performance among all methods. While PPO underperforms both in CR and SR relative to the other RL methods and even the Equal Weights benchmark, DDPG and SAC demonstrate strong potential, delivering higher returns with comparable or better drawdowns and volatility. Interestingly, the Equal and Random Weight baselines perform surprisingly well, underscoring the challenge of consistently outperforming naive strategies in real markets.

6 Conclusion

In this study, we explored the application of four reinforcement learning (RL) algorithms—A2C, DDPG, PPO, and SAC—for portfolio allocation across a diversified basket of stocks. Our approach extended beyond conventional financial indicators by incorporating company-level and sector-level sentiment signals as part of the feature set, enabling the agents to make more informed, context-aware trading decisions.

We benchmarked the RL-based strategies against standard alternatives, including an equally weighted portfolio, a randomly weighted portfolio, and the S&P 500 index. The results showed that DDPG and SAC achieved superior performance, with SAC delivering the highest cumulative return (49.1%), reflecting strong capital appreciation, and DDPG producing the highest Sharpe ratio (2.59), indicating better risk-adjusted returns. These outcomes suggest that off-policy methods leveraging continuous action spaces are particularly effective for this task—especially when enhanced with alternative data sources such as company and sector sentiment.

In contrast, PPO underperformed across most metrics. This may reflect challenges in its policy update mechanism or misalignment with the reward structure used, underlining the importance of environment design and algorithm fit in financial RL applications.

Overall, our findings indicate that RL-based portfolio optimization, when enhanced with sentiment-driven features, can outperform traditional allocation methods. Future work could expand this line of inquiry by incorporating transaction costs, slippage, and real-world execution constraints, as well as exploring the integration of news and macroeconomic indicators for even richer state representations.

7 Limitations and Ethical Statement

This project was conducted solely for academic and educational purposes as part of a graduate-level course. The methods and results presented are not intended to serve as financial advice or investment guidance. No real financial assets were traded during the course of this research, and the authors disclaim any responsibility for the use of this work in actual trading or financial decision-making contexts.

Furthermore, due to limited computational resources, certain methodological trade-offs were made, including restricted hyperparameter tuning and the use of a relatively small portfolio of stocks. These constraints may limit the generalizability of the findings and should be taken into account when interpreting the results.

References

[1] Mostapha Benhenda. 2025. FinRL-DeepSeek: LLM-Infused Risk-Sensitive Reinforcement Learning for Trading Agents. arXiv:2502.07393 [q-fin.TR] https://arxiv.org/abs/2502.07393

[2] Zihan Dong, Xinyu Fan, and Zhiyuan Peng. 2024. FNSPID: A Comprehensive Financial News Dataset in Time Series. arXiv:2402.06698 [q-fin.ST] https://arxiv.org/abs/2402.06698

[3] George Fatouros, Kostas Metaxas, John Soldatos, and Dimosthenis Kyriazis. 2024. Can Large Language Models beat wall street? Evaluating GPT-4's impact on financial decision-making with MarketSenseAI. *Neural Computing and Applications* (Dec. 2024). doi:10.1007/s00521-024-10613-4

[4] Jun GE, Yuanqi QIN, Yaling Li, yanjia Huang, and Hao Hu. 2022. Single stock trading with deep reinforcement learning: A comparative study. In *Proceedings of the 2022 14th International Conference on Machine Learning and Computing* (Guangzhou, China) *(ICMLC '22)*. Association for Computing Machinery, New York, NY, USA, 34–43. doi:10.1145/3529836.3529857

[5] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. arXiv:1801.01290 [cs.LG] https://arxiv.org/abs/1801.01290

[6] Gang Huang, Xiaohua Zhou, and Qingyang Song. 2025. Deep Reinforcement Learning for Long-Short Portfolio Optimization. arXiv:2012.13773 [q-fin.CP] https://arxiv.org/abs/2012.13773

[7] Ivan Letteri. 2023. VolTS: A Volatility-based Trading System to forecast Stock Markets Trend using Statistics and Machine Learning. arXiv:2307.13422 [q-fin.TR] https://arxiv.org/abs/2307.13422

[8] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2019. Continuous control with deep reinforcement learning. arXiv:1509.02971 [cs.LG] https://arxiv.org/abs/1509.02971

[9] Xiao-Yang Liu, Zhuoran Xiong, Shan Zhong, Hongyang Yang, and Anwar Walid. 2022. Practical Deep Reinforcement Learning Approach for Stock Trading. arXiv:1811.07522 [cs.LG] https://arxiv.org/abs/1811.07522

[10] Alejandro Lopez-Lira and Yuehua Tang. 2024. Can ChatGPT Forecast Stock Price Movements? Return Predictability and Large Language Models. arXiv:2304.07619 [q-fin.ST] https://arxiv.org/abs/2304.07619

[11] Ben R. Marshall and Martin Young. 2003. Liquidity and stock returns in pure order-driven markets: evidence from the Australian stock market. *International Review of Financial Analysis* 12, 2 (2003), 173–188. https://ideas.repec.org/a/eee/finana/v12y2003i2p173-188.html

[12] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous Methods for Deep Reinforcement Learning. arXiv:1602.01783 [cs.LG] https://arxiv.org/abs/1602.01783

[13] Ashish Anil Pawar, Vishnureddy Prashant Muskawar, and Ritesh Tiku. 2024. Portfolio Management using Deep Reinforcement Learning. arXiv:2405.01604 [q-fin.PM] https://arxiv.org/abs/2405.01604

[14] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. 2021. Stable-baselines3: reliable reinforcement learning implementations. *J. Mach. Learn. Res.* 22, 1, Article 268 (Jan. 2021), 8 pages.

[15] Rafał Rak, Stanisław Drożdż, Jarosław Kwapień, and Paweł Oświecimka. 2015. Detrended cross-correlations between returns, volatility, trading activity, and volume traded for the stock market companies. *EPL (Europhysics Letters)* 112, 4 (Nov. 2015), 48001. doi:10.1209/0295-5075/112/48001

[16] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347 [cs.LG] https://arxiv.org/abs/1707.06347

[17] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. 2014. Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32* (Beijing, China) *(ICML'14)*. JMLR.org, I–387–I–395.

[18] Meiyun Wang, Kiyoshi Izumi, and Hiroki Sakaji. 2024. LLMFactor: Extracting Profitable Factors through Prompts for Explainable Stock Movement Prediction. arXiv:2406.10811 [cs.CL] https://arxiv.org/abs/2406.10811

[19] Yijia Xiao, Edward Sun, Di Luo, and Wei Wang. 2025. TradingAgents: Multi-Agents LLM Financial Trading Framework. arXiv:2412.20138 [q-fin.TR] https://arxiv.org/abs/2412.20138

[20] Guojun Xiong, Zhiyang Deng, Keyi Wang, Yupeng Cao, Haohang Li, Yangyang Yu, Xueqing Peng, Mingquan Lin, Kaleb E Smith, Xiao-Yang Liu, Jimin Huang, Sophia Ananiadou, and Qianqian Xie. 2025. FLAG-Trader: Fusion LLM-Agent with Gradient-based Reinforcement Learning for Financial Trading. arXiv:2502.11433 [cs.AI] https://arxiv.org/abs/2502.11433

[21] Yi Yang, Mark Christopher Siy UY, and Allen Huang. 2020. FinBERT: A Pretrained Language Model for Financial Communications. arXiv:2006.08097 [cs.CL] https://arxiv.org/abs/2006.08097

[22] Chengyang Ying, Xinning Zhou, Hang Su, Dong Yan, Ning Chen, and Jun Zhu. 2022. Towards Safe Reinforcement Learning via Constraining Conditional Value-at-Risk. arXiv:2206.04436 [cs.LG] https://arxiv.org/abs/2206.04436

[23] Yangyang Yu, Haohang Li, Zhi Chen, Yuechen Jiang, Yang Li, Denghui Zhang, Rong Liu, Jordan W. Suchow, and Khaldoun Khashanah. 2023. FinMem: A Performance-Enhanced LLM Trading Agent with Layered Memory and Character Design. arXiv:2311.13743 [q-fin.CP] https://arxiv.org/abs/2311.13743

[24] Yangyang Yu, Zhiyuan Yao, Haohang Li, Zhiyang Deng, Yupeng Cao, Zhi Chen, Jordan W. Suchow, Rong Liu, Zhenyu Cui, Zhaozhuo Xu, Denghui Zhang, Koduvayur Subbalakshmi, Guojun Xiong, Yueru He, Jimin Huang, Dong Li, and Qianqian Xie. 2024. FinCon: A Synthesized LLM Multi-Agent System with Conceptual Verbal Reinforcement for Enhanced Financial Decision Making. arXiv:2407.06567 [cs.CL] https://arxiv.org/abs/2407.06567

[25] Chong Zhang, Xinyi Liu, Zhongmou Zhang, Mingyu Jin, Lingyao Li, Zhenting Wang, Wenyue Hua, Dong Shu, Suiyuan Zhu, Xiaobo Jin, Sujian Li, Mengnan Du, and Yongfeng Zhang. 2024. When AI Meets Finance (StockAgent): Large Language Model-based Stock Trading in Simulated Real-world Environments. arXiv:2407.18957 [q-fin.TR] https://arxiv.org/abs/2407.18957

A Appendix

A.1 Deep Reinforcement Learning Algorithms

This section contains pseudocode for the deep reinforcement learning algorithms used (PPO, DDPG, SAC and A2C)

Algorithm 1 Proximal Policy Optimization

Input: initial policy parameters θ_0, initial value function parameters ϕ_0
for $k = 0, 1, 2, \ldots$ **do**
  Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
  Compute rewards-to-go $\hat{R}_t$.
  Compute advantage estimates $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.
  Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} \hat{A}_t^{\pi_{\theta_k}}(s_t, a_t),\ g(\epsilon, \hat{A}_t^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

    typically via stochastic gradient ascent with Adam.
  Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

    typically via some gradient descent algorithm.
end for

Algorithm 2 Deep Deterministic Policy Gradient

Input: initial policy parameters θ, Q-function parameters ϕ, empty replay buffer $\mathcal{D}$
Set target parameters equal to main parameters $\theta_{\text{targ}} \leftarrow \theta$, $\phi_{\text{targ}} \leftarrow \phi$
repeat
  Observe state s and select action $a = \text{clip}(\mu_\theta(s) + \epsilon, a_{\text{Low}}, a_{\text{High}})$, where $\epsilon \sim \mathcal{N}$
  Execute a in the environment
  Observe next state s', reward r, and done signal d to indicate whether s' is terminal
  Store (s, a, r, s', d) in replay buffer $\mathcal{D}$
  if s' is terminal **then**
    Reset environment state
  end if
  if it's time to update **then**
    for however many updates **do**
      Randomly sample a batch of transitions, $B = \{(s, a, r, s', d)\}$ from $\mathcal{D}$
      Compute targets:
$y(r, s', d) = r + \gamma(1 - d) Q_{\phi_{\text{targ}}}(s', \mu_{\theta_{\text{targ}}}(s'))$
      Update Q-function by one step of gradient descent using
$\nabla_\phi \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} (Q_\phi(s, a) - y(r, s', d))^2$
      Update policy by one step of gradient ascent using
$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} Q_\phi(s, \mu_\theta(s))$
      Update target networks with
$\phi_{\text{targ}} \leftarrow \rho \phi_{\text{targ}} + (1 - \rho)\phi$
$\theta_{\text{targ}} \leftarrow \rho \theta_{\text{targ}} + (1 - \rho)\theta$
    end for
  end if
until convergence

Algorithm 3 Soft Actor-Critic

Input: initial policy parameters θ, Q-function parameters ϕ_1, ϕ_2, empty replay buffer $\mathcal{D}$
Set target parameters equal to main parameters $\phi_{\text{target},1} \leftarrow \phi_1$, $\phi_{\text{target},2} \leftarrow \phi_2$
repeat
  Observe state s and select action $a \sim \pi_\theta(\cdot|s)$
  Execute a in the environment
  Observe next state s', reward r, and done signal d to indicate whether s' is terminal
  Store (s, a, r, s', d) in replay buffer $\mathcal{D}$
  if s' is terminal **then**
    Reset environment state
  end if
  if it's time to update **then**
    for j in range (however many updates) **do**
      Randomly sample a batch of transitions, $B = \{(s, a, r, s', d)\}$ from $\mathcal{D}$
      Compute targets for the Q functions:

$$y(r, s', d) = r + \gamma(1 - d)\left(\min_{i=1,2} Q_{\phi_{\text{target},i}}(s', \tilde{a}') - \alpha \log \pi_\theta(\tilde{a}'|s')\right), \quad \tilde{a}' \sim \pi_\theta(\cdot|s')$$

      Update Q-functions by one step of gradient descent using:

$$\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} \left(Q_{\phi_i}(s, a) - y(r, s', d)\right)^2 \quad \text{for } i = 1, 2$$

      Update policy by one step of gradient ascent using:

$$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} \left(\min_{i=1,2} Q_{\phi_i}(s, \tilde{a}_\theta(s)) - \alpha \log \pi_\theta(\tilde{a}_\theta(s)|s)\right),$$

      where $\tilde{a}_\theta(s)$ is a sample from $\pi_\theta(\cdot|s)$ which is differentiable w.r.t. θ via the reparameterization trick.
      Update target networks with:

$$\phi_{\text{target},i} \leftarrow \rho \phi_{\text{target},i} + (1 - \rho)\phi_i \quad \text{for } i = 1, 2$$

    end for
  end if
until convergence

Algorithm 4 Advantage Actor-Critic A2C

Assume parameter vectors θ **and** θ_v
Initialize step counter $t \leftarrow 1$
Initialize episode counter $E \leftarrow 1$
repeat
  Reset gradients: $d\theta \leftarrow 0$ and $d\theta_v \leftarrow 0$
  $t_{\text{start}} \leftarrow t$
  Get state s_t
  repeat
    Perform a_t according to policy $\pi(a_t|s_t;\theta)$
    Receive reward r_t and new state s_{t+1}
    $t \leftarrow t+1$
  until terminal s_t or $t - t_{\text{start}} == t_{\text{max}}$
  $R \leftarrow \begin{cases} 0 & \text{for terminal } s_t \\ V(s_t,\theta_v) & \text{for non-terminal } s_t \text{ // Bootstrap from last state} \end{cases}$
  for $i \in \{t-1,\ldots,t_{\text{start}}\}$ **do**
    $R \leftarrow r_i + \gamma R$
    Accumulate gradients wrt θ:
      $d\theta \leftarrow d\theta + \nabla_\theta \log \pi(a_i|s_i;\theta)(R - V(s_i;\theta_v)) + \beta_e \partial H(\pi(a_i|s_i;\theta))/\partial\theta$
    Accumulate gradients wrt θ_v:
      $d\theta_v \leftarrow d\theta_v + \beta_v(R - V(s_i;\theta_v))(\partial V(s_i;\theta_v)/\partial\theta_v)$
  end for
  Perform update of θ using $d\theta$ and of θ_v using $d\theta_v$
  $E \leftarrow E+1$
until $E > E_{\text{max}}$

A.2 Hyperparameters

This section presents the hyperparameter settings used for each deep reinforcement learning algorithm after manual tuning. For each method (PPO, DDPG, SAC and A2C), we evaluated several combinations of parameters including learning rate, batch size, and number of steps per update, selecting the configuration that yielded the best validation performance in terms of cumulative return and Sharpe ratio. These hyperparameters correspond to the implementation of Stable-Baselines3 [14] so the hyperparameter names might not correspond exactly with the pseudocode provided above.

Parameter	Value
learning_rate	5e-4
n_steps	4096
batch_size	256
n_epochs	10
gamma	0.99
gae_lambda	0.95
clip_range	0.2
ent_coef	0.01
vf_coef	0.05
total_timesteps	409600

Table 3: Best Hyperparameters for PPO

Parameter	Value
learning_rate	5e-4
buffer_size	50000
batch_size	256
learning_starts	100
gamma	0.95
tau	0.01
train_freq	2
gradient_steps	1
total_timesteps	409600

Table 4: Best Hyperparameters for DDPG

Parameter	Value
learning_rate	5e-4
buffer_size	1000000
batch_size	128
learning_starts	100
gamma	0.98
tau	0.01
train_freq	1
gradient_steps	2
total_timesteps	409600

Table 5: Best Hyperparameters for SAC

Parameter	Value
learning_rate	3e-4
n_steps	20
gamma	0.99
gae_lambda	0.95
ent_coef	0.01
vf_coef	0.5
total_timesteps	409600

Table 6: Best Hyperparameters for A2C