StockSense: Multi-Target Trading Automation with Deep Reinforcement Learning

Yan-Fu Chen
Dept. of Computer Science
National Tsing-Hua University
Hsinchu, Taiwan
aaaronyanfu@gmail.com

Yi-Ning Chang
Dept. of Computer Science
National Tsing-Hua University
Hsinchu, Taiwan
changyn@gapp.nthu.edu.tw

Sheng-You Chien
Dept. of Computer Science
National Tsing-Hua University
Hsinchu, Taiwan
s99086tobby@gmail.com

Bo-Yi Mao
Dept. of Computer Science
National Tsing-Hua University
Hsinchu, Taiwan
dogeon188@gapp.nthu.edu.tw

Jie-Hung Chen
Dept. of Computer Science
National Tsing-Hua University
Hsinchu, Taiwan
jiehongchen726@gmail.com

Yi-Hsueh Chu
Dept. of Computer Science
National Tsing-Hua University
Hsinchu, Taiwan
ethan111062332@gapp.nthu.edu.tw

***Abstract*—In this paper, we leverage deep reinforcement learning (DRL) to automate multi-target trading, where the goal is to optimize trading decisions across a portfolio of assets. We propose a flexible and extensible architecture that supports multiple stocks, various trading strategies, and different reinforcement learning algorithms. We evaluate the performance of our models on a real-world dataset and compare them with traditional buy-and-hold strategies.**

Our results show that our DRL models outperform the buy-and-hold strategy in terms of annualized return and cumulative return. However, the DRL models exhibit higher risk, which demonstrates the trade-off between return and risk in multi-target trading.

***Index Terms*—computational finance, deep reinforcement learning, algorithmic trading, multi-target trading**

I. Introduction

An automatic, algorithmic way of stock trading, leveraging the swiftness and accuracy of computer systems, has long been a persuit of many. With the rise of machine learning and deep learning, the possibility of using these technologies to predict stock prices and make trading decisions has become a reality.

Many studies have tackled the problem of stock price prediction using machine learning models, such as Support Vector Machines (SVMs), Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs). These models have shown promising results in forecasting stock prices, capturing temporal dependencies, and extracting relevant features from raw data.

However, few works have made a formal attempt to combine algorithmic trading, deep reinforcement learning, and multi-target trading. In this paper, we aim to address this gap by summarizing and extending the current state-of-the-art algorithms and methodologies for multi-target trading automation using deep reinforcement learning, and comparing their performance on a real-world dataset.

II. Related Works

Even though the Efficient Market Hypothesis (EMH) posits that stock prices reflect all available information, and thus cannot be predicted consistently, many studies have shown that stock prices exhibit predictable patterns and can be forecasted to some extent. Various approaches have been proposed, differing in learning algorithms and objectives. In recent years, deep reinforcement learning (DRL) has gained significant attention due to its ability to learn complex trading strategies directly from market data.

A. Stock Price Prediction

Most studies have focused on predicting stock prices using various machine learning models. Traditional statistical methods such as ARIMA and GARCH have been widely used for time series forecasting. However, financial time series are often non-stationary and exhibit volatility changes, making it challenging to capture their underlying patterns with smoothened models.

With the advent of machine learning and deep learning, more sophisticated models have been developed. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), including Long Short-Term Memory (LSTM) networks, have shown promising results in stock price prediction. These models can automatically extract relevant features from raw data and capture temporal dependencies, making them well-suited for financial time series analysis.

Support Vector Machines (SVMs) have been applied to stock predictions in many works. Phayung et al. [1] applied Support Vector Regression (SVR) and combines different windowing functions to predict stock prices. Their model showed good performance in predicting short-term stock prices. Similarly, Tripathy [2] demonstrated that SVMs are effective in highly volatile markets, and it performs better in short-term and medium-term predictions compared to long-term predictions. Also, Zhang et al. [3] proposed a periodic updating SVM model that can adapt to changing market conditions, which showed competitive performance compared to the random walk model proposed by the EMH.

Zhang et al. [3] proposed a State Frequency Memory (SFM) recurrent neural network to capture the multi-frequency trading patterns in stock prices. They demonstrated that their model showed competitive performance compared to traditional methods. Fischer and Krauss [4] applied LSTM networks to rank stocks based on their future returns and showed that their model could outperform traditional ranking methods.

More advanced models have also been applied to stock price prediction in recent years. Feng et al. [5] proposed an adversarial learning framework that outperformed SOTA methods in terms of prediction accuracy. Koa et al. [6] developed a combination of Variational Autoencoders (VAEs) and diffusion models to capture the latent structure of financial time series. Gao et al. [7] proposed a transformer-based model that outperformed traditional methods in terms of portfolio returns and Sharpe ratios.

B. Algorithmic Trading

Deep reinforcement learning (DRL), a branch of machine learning where an agent learns to make decisions by interacting with an environment to maximize cumulative rewards, is capable of learning complex trading strategies directly from market data, making it an attractive approach for algorithmic trading. For example, Deep Q-Learning (DQN) has been widely applied to stock trading for decision-making in dynamic and uncertain environments. Jiang et al. [8] developed a deep reinforcement learning model that can adapt to changing market conditions and showed competitive performance compared to traditional methods. Liu et al. [9] proposed a deep reinforcement learning library, FinRL, tailored for financial markets, enabling multi-stock trading strategies. Their library provides a comprehensive set of tools for developing and evaluating trading strategies using reinforcement learning, including various environments, algorithms, and evaluation metrics.

Despite these advancements in stock price prediction, they are usually evaluated on historical data and do not consider the trading strategy's performance in real-world scenarios. few works have addressed the challenges of multi-target trading, where the goal is to optimize trading decisions across a portfolio of assets. Multi-target trading involves managing multiple stocks simultaneously, considering their interdependencies and correlations to maximize overall portfolio returns.

One notable work in this area is by Feng et al. [10], who proposed an improved ranking model that can generate trading signals based on the predicted stock returns. Importantly, they considered the relationship between different stocks that conveys valuable information for trading decisions.

III. Methodology

The utilized multi-target trading architecture is designed to handle multiple stock assets simultaneously, optimizing trading decisions across a portfolio rather than focusing on a single stock. To address this challenge, we leverage DRL algorithms with a flexible environment that supports multi-asset trading. Below, we outline the key components of the architecture:

A. Reinforcement Learning Overview

Reinforcement learning is based on the interaction between an **agent** and an **environment**. The agent observes the state of the environment, takes actions, and receives rewards based on the results of those actions. The goal of the agent is to learn an optimal policy that maximizes the cumulative reward over time.

- **State** (S_t): Represents the information observed by the agent from the environment at time t.
- **Action** (A_t): The decision made by the agent at time t, such as buying, selling, or holding stocks.
- **Reward** (R_t): A scalar feedback signal that evaluates the action taken by the agent in a specific state.
- **Policy** (π): A mapping from states to actions, guiding the agent's decision-making process.
- **Q-Function** ($Q(s, a)$): Represents the expected cumulative reward of taking action a in state s and following the policy thereafter.

We use advanced RL algorithms such as *Deep Q-Network (DQN)*, *Advantage Actor-Critic (A2C)*, and *Proximal Policy Optimization (PPO)* to train our agents.

B. Environment

The environment simulates a stock market, incorporating realistic features such as price changes, transaction costs, and investment constraints. It provides the agent with state information derived from historical market data, enabling the agent to learn and make trading decisions.

- **Data Source**: The environment accepts a list of Pandas DataFrames containing historical stock price data for multiple assets. For each DataFrame, the columns represent the OHLCV data (open, high, low, close prices, and trading volume) at each time step.
- **Implementation**: The environment is implemented by extending the Gymnasium `Env` class, allowing seamless integration with reinforcement learning frameworks.

C. State Representation

The state representation is critical for effective learning. Since raw stock market data varies greatly in scale and lacks direct interpretability, we preprocess and normalize it into meaningful features:

- **Normalized Price Features**:
$$\text{Open\%} = \frac{\text{Open}}{\text{Close}}, \quad \text{High\%} = \frac{\text{High}}{\text{Close}}, \quad \text{Low\%} = \frac{\text{Low}}{\text{Close}}$$
- **Percentage Change of Close**: Measures the relative change in the close price compared to the previous day.
- **Normalized Volume**: Volume as a percentage of the maximum trading volume within a specified time window (100 days).

The agent observes the last five days of these features, creating a sliding window of information to capture short-term trends.

D. Actions

The action space defines how the agent interacts with the environment. In our implementation, we support two modes of trading actions:

1) **All-In Mode**: The agent selects a single stock to allocate 100% of its capital.
2) **Proportional Allocation Mode**: The agent determines the percentage of capital to allocate across multiple stocks, enabling diversified strategies.

This flexibility allows the agent to explore both high-risk and balanced investment strategies.

E. Reward Function

We use the **logarithmic gain ratio** as the reward function to evaluate the performance of the agent's decisions. This reward structure emphasizes proportional gains, penalizes losses, and encourages consistent growth. The reward for a given action is calculated as:

$$R_t = \log\left(\frac{\text{Portfolio Value}_t}{\text{Portfolio Value}_{t-1}}\right)$$

This formulation accounts for transaction costs and ensures that the agent focuses on maximizing net returns.

F. Agent Design

To solve the trading problem effectively, we utilize three reinforcement learning algorithms:

- **Deep Q-Network (DQN)**: Uses a neural network to approximate the Q-function. It is particularly effective for discrete action spaces and leverages experience replay and target networks for stability.
- **Advantage Actor-Critic (A2C)**: A policy-gradient method that combines value-based and policy-based approaches. It learns both a value function (critic) and a policy (actor) simultaneously, enabling efficient exploration.
- **Proximal Policy Optimization (PPO)**: A robust and scalable policy-gradient method. It improves training stability by constraining policy updates to avoid drastic changes, ensuring better convergence.

Each algorithm is trained to optimize portfolio returns while managing risks and transaction costs.

IV. Experiments

A. Dataset

We obtained the stock price data from Yahoo! Finance [11] using the yfinance Python library [12], and preprocess the data to fit the requirements of our trading environment. The dataset used in our experiments includes the daily OHLCV data for the following stocks: Microsoft (MSFT), Apple (AAPL), American Express (AXP), Amgen (AMGN), Cisco Systems (CSCO), Honeywell (HON), IBM (IBM), and Coca-Cola (KO). It contains data points spanning from January 1, 2000 to December 1, 2024.

The dataset is split into three parts:

- Training Data: First 70% of the data (from January 1, 2000 to June 12, 2017).
- Validation Data: Next 10% of the data (from June 13, 2017 to December 6, 2019).
- Testing Data: Last 20% of the data (from December 7, 2019 to December 1, 2024).

The training data is used to train the reinforcement learning models, while the validation data is used to tune hyperparameters and prevent overfitting. The testing data is used to evaluate the performance of the trained models.

To visualize the historical stock prices used in our experiments, we plot the closing prices of the selected stocks over the entire dataset period in Figure 1.

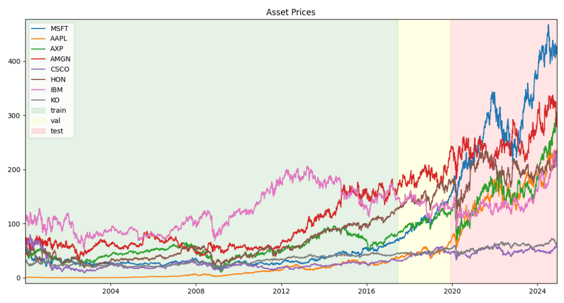

Fig. 1. Historical closing prices of selected stocks from January 1, 2000 to December 1, 2024.

B. Evaluation Metrics

Backtesting is a common method to evaluate the performance of a predictive model in the financial domain. It simulates the trading strategy on historical data to estimate the strategy's performance.

We use FinRL [9], a powerful deep reinforcement learning library to perform backtest on our trading strategy. Our reinforcement learning model is compared with the buy-and-hold (B&H) strategy, which buys and holds the stock for the entire trading period, on the DJI Index. DJI Index is a stock market index that measures the stock performance of 30 large companies listed on stock exchanges in the United States.

Our models are evaluated based on the following metrics:

- Annualized Return: The annualized return of the trading strategy.
- Cumulative Return: The cumulative return of the trading strategy.
- Maximum Drawdown: The maximum loss from a peak (highest point) to a trough (lowest point) of a trading strategy. A higher maximum drawdown indicates the trading strategy has higher risk.
- Annualized Volatility: The annualized volatility is defined as the standard deviation of the asset's yearly logarithmic

return. A higher annualized volatility indicates the trading strategy has higher risk.

- Sharpe Ratio: As measure of the risk-adjusted return of the trading strategy, the Sharpe ratio is defined as:

$$\text{Sharpe Ratio} = \frac{E[R_p - R_f]}{\sigma_p} \tag{1}$$

where R_p is the asset return of the trading strategy, σ_p is the standard deviation of the asset excess return, and R_f is the risk-free return (such as a U.S. Treasury security). Therefore, The Sharpe ratio characterizes how well the return of an asset compensates the investor for the risk taken. A higher Sharpe ratio indicates the strategy can generate higher returns with lower risk.

- Calmar Ratio: The Calmar ratio is defined as the ratio of the annualized return to the maximum drawdown of the trading strategy:

$$\text{Calmar Ratio} = \frac{\text{Annualized Return}}{\text{Maximum Drawdown}} \tag{2}$$

A higher Calmar ratio indicates the strategy can generate higher returns with lower downside risk. Compared to the Sharpe ratio, the Calmar ratio focuses on the downside risk of the trading strategy.

- Sortino Ratio: Sortino ratio is a variation of the Sharpe ratio that only considers the downside risk of the trading strategy. It is defined as:

$$\text{Sortino Ratio} = \frac{E[R_p - R_f]}{\sigma_d} \tag{3}$$

where σ_d is the standard deviation of the asset downside return. A higher Sortino ratio indicates the strategy can generate higher returns with lower downside risk.

- Omega Ratio: The Omega ratio is a risk-return measure that evaluates the probability of achieving a certain return. It is defined as:

$$\text{Omega Ratio} = \frac{P(R_p > R_f)}{P(R_p < R_f)} \tag{4}$$

where $P(R_p > R_f)$ is the probability of achieving a return greater than the risk-free rate, and $P(R_p < R_f)$ is the probability of achieving a return less than the risk-free rate. A higher Omega ratio indicates the strategy has a higher probability of achieving a return greater than the risk-free rate.

C. Results

This section presents the experimental results of applying different models and strategies to stock market prediction and portfolio management. The model we implemented in this study is the Deep Q-Network (DQN), and it is compared against several baseline strategies.

1) Asset Prices Over Time: Figure 1 shows the historical performance of selected stocks, including MSFT, AAPL, AXP, AMGN, CSCO, HON, IBM, and KO. The plot is segmented into three periods: the training period (green background), the validation period (yellow background), and the test period (pink background).

From the chart, it is evident that AAPL and MSFT exhibit the most significant growth trajectories, particularly during the test period, where their values rise steeply. MSFT surpasses all other stocks in terms of cumulative performance, highlighting its dominance in the observed timeframe. This exceptional growth could be attributed to factors such as market innovation, strong earnings, or favorable macroeconomic conditions. Conversely, stocks like KO and IBM display relatively flat trajectories, indicating limited growth potential or consistent performance without major volatility.

AMGN demonstrates a notable rise during the validation period but appears to plateau in the test period, which could reflect sector-specific trends or external market influences that hindered its continued growth. AXP, on the other hand, shows moderate growth but remains significantly behind the tech-driven advancements of AAPL and MSFT, underscoring the variability in performance across different industries.

2) Model Performance Comparison: Table I and Figure 2 present the performance comparison of the B&H baseline against DRL strategies, including DQN, A2C, and PPO. The evaluation metrics include annualized return, cumulative return, maximum drawdown, Sharpe ratio, Calmar ratio, annualized volatility, and Sortino ratio.

- **Cumulative Returns:** The DQN model achieves the highest cumulative returns among all strategies, significantly outperforming traditional approaches like B&H. This demonstrates the DQN's capability to dynamically adjust portfolio allocations and capitalize on market inefficiencies.
- **Sharpe Ratio:** The Sharpe ratio for DQN is notably higher than that of most baseline strategies, indicating superior risk-adjusted returns. This metric highlights the model's ability to balance returns against market volatility effectively.
- **Maximum Drawdown:** Despite its strengths, the DQN model exhibits a relatively large maximum drawdown. This underscores its susceptibility to sharp market downturns, potentially due to overfitting or high sensitivity to volatile conditions.
- **Strategy Observations:** For the tested strategies, the following observations are made:
  - **B&H:** This strategy delivers stable returns with a relatively higher Sharpe ratio compared to other baseline models. While it minimizes risk and avoids major drawdowns, its lack of adaptability prevents it from taking full advantage of upward market trends, resulting in moderate cumulative returns.
  - **DQN:** DQN achieves the highest cumulative returns and Sharpe ratio among all strategies, indicating its superior performance in optimizing portfolio allocations and managing risk. However, its larger maximum drawdown suggests potential vulnerabilities to sudden market downturns or overfitting to specific market conditions.

- **A2C:** A2C achieves moderate performance with higher cumulative returns than PPO but struggles to match the adaptability of more sophisticated models. Its Sharpe ratio indicates balanced risk-adjusted returns, though it remains lower than that of DQN.
- **PPO:** Although PPO demonstrates some capability in managing risk, its cumulative returns and Sharpe ratio are the lowest among the three. This suggests that PPO failed to effectively capture upward market trends during the test period and did not outperform the other baseline strategies in balancing risk and reward

TABLE I
PERFORMANCE COMPARISON OF THE STRATEGIES ON OUR DATASET.

Evaluation Metric	**Model**			
	DQN	***A2C***	***PPO***	***Buy-and-Hold***
Annualized Return	<u>0.3122</u>	0.2119	0.0192	0.2387
Cumulative Return	<u>0.3969</u>	0.2667	0.0237	0.3001
Maximum Drawdown	-0.1807	-0.1549	-0.2074	<u>-0.0678</u>
Sharpe Ratio	1.4614	1.0595	0.1955	<u>2.1126</u>
Calmar Ratio	1.7389	1.3682	0.0925	<u>3.5230</u>
Annualized Volatility	0.1999	0.2012	0.1941	<u>0.1043</u>
Sortino Ratio	1.5270	1.4550	0.2661	<u>3.2108</u>

Note: The best performance for each metric is underlined.

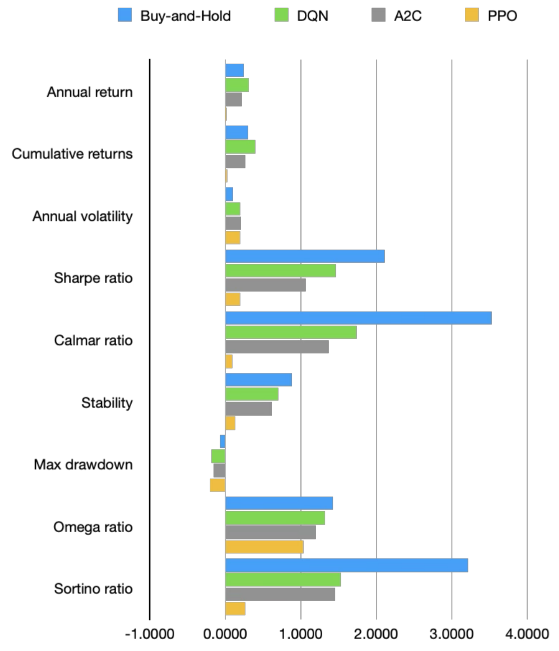

Fig. 2. Performance comparison of the strategies on our dataset.

V. Discussion

This section analyzes the experimental results and provides potential insights and improvements for the DQN model based on its performance.

A. Analysis of Results

- **DQN's Strengths:** The exceptional cumulative returns achieved by the DQN model stem from its use of deep Q-learning, which enables it to efficiently learn optimal policies for asset allocation by mapping states to value estimates. This adaptability enables the model to exploit trends in high-performing assets, such as AAPL and MSFT, particularly during periods of rapid growth. Furthermore, the consistently high Sharpe ratio demonstrates the model's capacity to manage risk effectively while maximizing returns.
- **Drawdown Challenges:** Despite its strong performance metrics, the DQN model's larger maximum drawdown compared to baseline strategies suggests a potential vulnerability to sudden market downturns. This may arise from overfitting to specific market conditions during training or an over-reliance on short-term signals that increase risk exposure.

B. Potential Improvements

- **Regularization and Robustness:** To address overfitting, implementing regularization techniques, such as dropout layers or weight decay, could improve the model's generalization capabilities. Additionally, ensemble methods could provide more stable performance across different market conditions by reducing the impact of outliers.
- **Risk Management:** Incorporating explicit risk management objectives, such as drawdown constraints or volatility penalties, into the reward function could help mitigate excessive risk exposure. By aligning the model's behavior with practical investment strategies, these enhancements could improve its robustness in real-world applications.
- **Feature Engineering:** Expanding the feature set with macroeconomic indicators, such as interest rates or inflation data, and alternative data sources, such as sentiment analysis from news or social media, could improve the model's predictive capabilities. Additionally, incorporating sector-specific indicators may help capture trends that are unique to particular industries.
- **Dynamic Reward Functions:** Modifying the reward function to account for long-term performance metrics, such as capital preservation or reduced portfolio turnover, could encourage more sustainable and consistent investment strategies. This would also address concerns regarding high transaction costs associated with frequent portfolio rebalancing.

In conclusion, while the DQN model demonstrates significant potential in outperforming traditional portfolio management strategies, careful consideration of its limitations, particularly regarding risk and robustness, is crucial for practical deployment. Future research could explore the integration of multi-agent reinforcement learning frameworks or hybrid approaches to enhance the model's adaptability and performance in complex and dynamic financial markets.

VI. Conclusion

In this paper, we proposed a flexible and extensible architecture for multi-target trading automation using deep reinforcement learning. We evaluated several reinforcement learning algorithms, including DQN, A2C, and PPO, on a real-world dataset containing historical stock price data for multiple assets.

Our results show that our DRL models outperform the baseline buy-and-hold strategy in terms of annualized return and cumulative return. However, the DRL models exhibit higher risk, as indicated by the maximum drawdown and annualized volatility. As our model is optimized for return, such risky behavior is expected. This demonstrates the trade-off between return and risk in multi-target trading.

The forementioned results also suggest that our models may benefit from additional risk management techniques to reduce downside risk and improve overall performance. With a more sophisticated reward function that considers risk-adjusted returns, we can potentially achieve a better balance between return and risk. Furthermore, incorporating additional features such as technical indicators, market sentiment, and macroeconomic factors could enhance the model's predictive power and robustness. Future work could explore these directions to further improve the performance of our models.

VII. Data and Code Availability

Stock price data used in this study is publicly available from Yahoo! Finance [11], and is obtained using the yfinance Python library [12]. Cryptocurrency price data is obtained from the Binance API [13] using the ccxt Python library [14].

All code for data fetching, data preprocessing, model training, and evaluation is available at https://github.com/Dogeon188/StockSense. Any updates or changes to the code will be reflected in the GitHub repository.

VIII. Author Contribution Statements

- **Y. Chen** *(16.9%)*: study research, methodology design.
- **Y. Chang** *(16.7%)*: result analysis, proposal presentation, evaluation metrics.
- **B. Mao** *(16.7%)*: data curation, video presentation, project outline.
- **S. Chien** *(16.5%)*: methodology design, result analysis.
- **J. Chen** *(16.5%)*: methodology design, evaluation metrics.
- **Y. Chu** *(16.5%)*: study research, data curation.

References

[1] P. Meesad and R. I. Rasel, "Predicting stock market price using support vector regression," in *2013 International Conference on Informatics, Electronics and Vision (ICIEV)*, 2013, pp. 1–6.

[2] N. Tripathy, "Stock price prediction using support vector machine approach," in *International Academic Conference on Management & Economics*, 2019, pp. 44–59.

[3] L. Zhang, C. Aggarwal, and G.-J. Qi, "Stock price prediction via discovering multi-frequency trading patterns," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 2141–2149. [Online]. Available: https://doi.org/10.1145/3097983.3098117

[4] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *European Journal of Operational Research*, vol. 270, no. 2, pp. 654–669, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221717310652

[5] F. Feng, H. Chen, X. He, J. Ding, M. Sun, and T.-S. Chua, "Enhancing stock movement prediction with adversarial training," 2019. [Online]. Available: https://arxiv.org/abs/1810.09936

[6] K. J. Koa, Y. Ma, R. Ng, and T.-S. Chua, "Diffusion variational autoencoder for tackling stochasticity in multi-step regression stock price prediction," in *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, ser. CIKM '23. ACM, Oct. 2023, p. 1087–1096. [Online]. Available: http://dx.doi.org/10.1145/3583780.3614844

[7] S. Gao, Y. Wang, and X. Yang, "Stockformer: Learning hybrid trading machines with predictive coding," in *IJCAI*, 2023.

[8] Z. Jiang, D. Xu, and J. Liang, "A deep reinforcement learning framework for the financial portfolio management problem," 2017. [Online]. Available: https://arxiv.org/abs/1706.10059

[9] X.-Y. Liu, H. Yang, Q. Chen, R. Zhang, L. Yang, B. Xiao, and C. D. Wang, "Finrl: A deep reinforcement learning library for automated stock trading in quantitative finance," 2022. [Online]. Available: https://arxiv.org/abs/2011.09607

[10] F. Feng, X. He, X. Wang, C. Luo, Y. Liu, and T.-S. Chua, "Temporal relational ranking for stock prediction," *ACM Transactions on Information Systems*, vol. 37, no. 2, p. 1–30, Mar. 2019. [Online]. Available: http://dx.doi.org/10.1145/3309547

[11] "Yahoo! finance," https://finance.yahoo.com/, accessed Dec. 23, 2024.

[12] "Yfinance," https://github.com/ranaroussi/yfinance, accessed Dec. 23, 2024.

[13] "Binance api," https://www.binance.com/binance-api, accessed Dec. 23, 2024.

[14] "ccxt," https://github.com/ccxt/ccxt, accessed Dec. 23, 2024.