Leveraging Transformer in Deep Reinforcement Learning for Portfolio Optimization

Qingyi Xia, Siting Chen, Zhifei Dou

Abstract

This article explores the use of Transformer models within a Deep Reinforcement Learning (DRL) framework for portfolio optimization. Inspired by the paper 'Deep Reinforcement Learning for Optimal Portfolio Allocation', by the JP Morgan AI Research Group, as financial markets grow increasingly complex, traditional methods, such as Mean-Variance Optimization (MVO), often struggle to adapt to dynamic market conditions. Also inspired by breakthroughs in deep learning and the transformative "*Attention Is All You Need*" paper, this work aims to leverage the unique capabilities of Transformers to enhance decision-making in portfolio management. The team hypothesized that Transformers can better capture the intricate temporal dependencies inherent in financial data. To test this hypothesis, the team employed a Transformer-based Actor-Critic model to simulate financial environments using real-world data from five prominent Chinese stocks. The approach is benchmarked against three baseline models: a quadratic programming-based optimizer, fully connected neural networks (ANN), and Long Short-Term Memory Networks (LSTM). The comparison is conducted to evaluate whether Transformers can outperform these established approaches in balancing risk and return while managing portfolio allocations in a dynamic financial context. The results indicate that while the Transformer-based DRL model is competitive, it does not significantly outperform the baseline models. This is likely due to limitations such as the simplified environment and the use of daily-level data, which restrict the model's ability to capture more intricate patterns within the financial dataset. The findings suggest that access to more detailed minute-level data and the incorporation of broader contextual features could further enhance the model's performance. This research contributes to advancing portfolio optimization by integrating Transformers with DRL, highlighting the adaptability of Transformers beyond Natural Language Processing (NLP) applications, and offering insights into the application of advanced AI techniques in financial decision-making.

GitHub: https://github.com/ZhifeiDou/MIE1666_RL_Portfolio_Optimization.git

1 Introduction

Effective financial resource management is a foundational aspect of modern investment strategies, with portfolio optimization playing a pivotal role in determining the optimal allocation of an investor's capital across diverse financial assets, such as cash, bonds, and equities, or within a single asset class, such as allocating available cash across selected stocks. The primary objective of portfolio optimization is to achieve a balance between maximizing returns and minimizing risks while aligning investments with specific financial goals and constraints. With the ever-growing complexity of financial markets, traditional methods like MVO frequently fall short in addressing the nuanced interdependencies between assets or responding swiftly to the rapidly evolving dynamics of market conditions.

Inspired by seminal works "Deep Reinforcement Learning for Optimal Portfolio Allocation" and "Attention Is All You Need" [1][2], this project investigates the application of Transformers within a DRL framework for portfolio optimization. Initially developed for NLP tasks, transformers are well-suited for sequential data analysis due to their advanced attention mechanisms, making them a promising tool for financial applications where temporal dependencies are prevalent. The objective of this study is to evaluate whether the distinctive features of

Transformers—such as multi-head attention and parallel processing—can enhance decision-making processes in portfolio management.

This investigation employs an Actor-Critic model integrated with Transformers to simulate a financial environment using real-world data from five Chinese stocks. The proposed approach is benchmarked against baseline models, including Quadratic Programming (QP), Artificial Neural Networks (ANN), and Long-Short Term Memory Networks (LSTM). Through this comparative analysis, the study aims to assess the efficacy of Transformer-based architectures in addressing the complexities of portfolio optimization. The findings not only contribute to the growing body of research on the use of DRL in financial contexts but also underscore the adaptability and versatility of Transformers beyond their traditional applications in NLP.

2 Related Work

The application of DRL in trading and portfolio management has been extensively explored in recent years, following the emergence of DRL as a prominent research area. A notable contribution is the paper '*Deep Reinforcement Learning for Optimal Portfolio Allocation*' by the J.P. Morgan AI Research Group, which reframes portfolio management as a sequential decision-making problem [1]. By introducing a differential Sharpe ratio, this approach allows for dynamic adaptation to evolving market conditions [3]. Leveraging the Actor-Critic framework, which simultaneously optimizes policy and value functions, DRL has shown substantial promise in improving management strategies. For instance, '*Cost-Sensitive Portfolio Selection via DRL*' incorporates transaction costs into the Actor-Critic model, demonstrating its flexibility, while studies like '*AlphaPortfolio: Direct Construction Through Deep Reinforcement Learning and Interpretable AI*' integrate DRL with attention-based neural networks, highlighting its capability to process diverse financial data effectively [4][5].

Parallel advancements in Transformers, first introduced in '*Attention Is All You Need*', have revolutionized sequential data processing in fields like NLP [2]. By utilizing multi-head attention mechanisms, Transformers excel at capturing long-term dependencies and analyzing complex temporal sequences, addressing limitations of recurrent models, which often struggle with efficiency and scalability when processing long sequences. The success of Transformers in finance has been demonstrated in studies such as '*Time Series Forecasting with Transformer Models and Application to Asset Management*', which showed their ability to model intricate temporal relationships in financial data [6].

Recently, integrating DRL with Transformer architectures has emerged as a promising approach to portfolio optimization. For instance, studies like '*Portfolio Management and Deep Learning*' illustrate how attention-based architectures can enhance the representation of financial time series for decision-making tasks [7]. This integration leverages the strengths of both techniques, combining the sequential decision-making capabilities of DRL with the ability of Transformers to process and analyze complex temporal patterns.

Building on these foundations, this project aims to integrate Transformers architectures into the Actor-Critic framework for portfolio optimization. By leveraging the attention mechanisms and parallel processing capabilities of Transformers, this approach addresses the sequential processing constraints of traditional models like LSTMs and fully connected networks. This framework enhances the ability to process and interpret financial time series data, making it particularly effective for high-frequency trading and long-term trend analysis.

3 Datasets

The team has chosen to focus on the Chinese stock market as the primary dataset source for this project, given its significance as one of the largest and most dynamic financial markets in the world. For the initial dataset, the team selected five industry-leading Chinese companies from key traditional sectors of the economy: Ping An Insurance (Group) Company of China, Industrial and Commercial Bank of China Limited, Kweichow Moutai Company Limited, Ping An Bank Company Limited, and PetroChina Company Limited. These companies were chosen for their market dominance, strong financial performance, and established presence in mature industries, which typically carry lower volatility and risks compared to emerging sectors [8].

Guided by the 'Data Selection Rationale' section in the paper from Ndikum et al, the team has selected and extracted the data, including stock prices and stock indices for each of the stocks from January 1, 2021, to August 29, 2024 [9]. (Please refer to Table 2 in Appendix A for all extracted data categories.)

However, there is substantial variance across the different data categories. For example, while the opening price of a stock may be around 40, its trading volume can reach hundreds of thousands. To address this disparity and ensure consistency across features, Min-Max normalization was applied, defined as:

$$X_{\text{scaled}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

The dataset has been divided into three sections based on time periods to facilitate training, validation, and testing. As shown in Figure 1, the training dataset spans from January 1, 2021, to June 30, 2023, while the validation dataset covers the period from July 1, 2023, to December 31, 2023. The testing dataset includes data from June 6, 2024, to August 29, 2024. To mimic the dataset split methodology used in Ndikum et al.'s work, a gap period from January 1, 2024, to May 31, 2024, has been introduced [9].

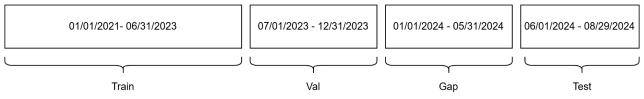

Figure 1. The Split of Dataset

4 Methods

4.1 Deep Reinforcement Learning (DRL)

Reinforcement Learning (RL) is a computational paradigm in which an agent learns to make sequential decisions through interactions with its environment, with the objective of maximizing cumulative rewards. This process is typically modeled using a Markov Decision Process (MDP) [10]. It consists of the following essential components [11][12]:

Agent: The decision-making entity that interacts with the environment.

Environment: The external system with which the agent interacts, encompassing all external factors and dynamics.

State(s): A representation of the environment's configuration at a specific time, encapsulating all relevant information.

Action(a): A set of possible decisions or moves the agent can execute from a given state.

Reward(r): A scalar feedback signal received after the agent takes an action, quantifying the immediate benefit or cost associated with that action.

Policy(π): A mapping from states to a probability distribution over actions, defining the agent's behavior strategy.

Value Function(V): Estimates the expected cumulative reward from a given state under a particular policy.

Action-Value Function(Q): Estimates the expected cumulative reward from taking a specific action in a given state under a particular policy.

DRL is a subfield of machine learning that combines RL principles with DL techniques. This integration enables the agent to make decisions from unstructured inputs, such as raw sensory data, by leveraging the representational power of deep neural networks.

4.2 Actor-Critic and Proximal Policy Optimization (PPO)

Actor-critic is an RL architecture originally introduced by Andrew et al in *'Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems'* [13]. This framework combines two key components:

Actor: A policy model that selects actions based on the current state.

Critic: A value model that evaluates the chosen actions by estimating their value, thus providing feedback to the actor. The high-level structure and flow of the actor-critic approach are illustrated in Figure 2.

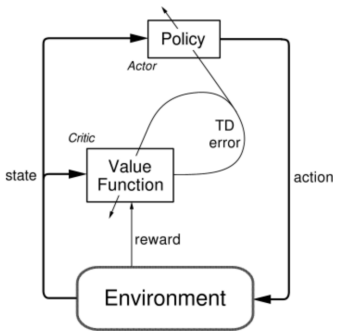

Figure 2. Structure of Actor-Critic Architecture

4.2.1 Actor (Policy Model):

The actor takes a given state s_t as input and outputs a probability distribution over possible actions a_t. Its primary objective is to maximize the expected return. The loss function for the actor, aiming to improve the policy, typically involves the advantage function $A(s_t|a_t)$, which quantifies the benefit of taking a specific action relative to the average expected outcome.

$$L_{\text{actor}} = -\mathbb{E}\left[A(s_t, a_t) \cdot \log \pi_\theta(a_t|s_t)\right]$$

Where:

$\pi_\theta(a_t|s_t)$: The actor's output that maps states to actions

$A(s_t|a_t)$: The advantage function, which measures how much better the selected action a_t is compared to

the average action at the state s_t is compared to an average action in that state.

4.2.2 Critic (Value Model):

The critic estimates the value of a state, typically denoted $V_\phi(s_t)$. The critic's loss function often takes the form of a mean squared error (MSE) between the predicted and target values:

$$L_{\text{critic}}(\phi) = \mathbb{E}\left[(V_\phi(s_t) - y_t)^2\right]$$

where V_ϕ is the value function parameterized by ϕ and y_t is the target value (often based on the next state's value plus the observed reward).

4.2.3 Temporal Difference (TD) Error:

In actor-critic, there is an essential concept of 'Temporal Difference (TD) error'. It quantifies the difference between the current value estimate and the better estimate. It can be expressed as:

$$\delta_t = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$$

Where:
r_t: Reward received at time step t.
γ: is the discount factor, and
$V_\phi(s_{t+1})$: the output of the value model, an estimate of the value of the state s_{t+1}.
$V_\phi(s_t)$: the output of the value model, an estimate of the value of the state s_t.

4.2.4 Proximal Policy Optimization (PPO):

Proximal Policy Optimization (PPO) is a specific actor-critic algorithm proposed in "Proximal Policy Optimization Algorithms" by John et al [14]. It ensures the policy does not change too drastically in a single update by introducing a clip loss function and penalty to limit large updates:

$$L^{CLIP}(\theta) = \mathbb{E}\left[\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)\right]$$

Where:
$r_t(\theta) = \frac{\pi_\theta(a\,|s\,)}{\pi_{\theta old}(a\,|s\,)}$ is the probability ratio between the current and previous policy model's output.

Unlike simpler actor-critic, PPO performs a number of epochs of optimization on a batch of data, with the clipping mechanism mentioned above ensuring the new policy doesn't significantly change from the last one. All PPO methods mentioned below will be simplified to the 'actor-critic' method for simplicity and consistency.

5 Experiment and Result

5.1 Experiment Setup

The team has implemented three baseline methods, which include the Quadratic programming method, the Fully Connected method, and the LSTM method. These baseline models will be compared with the proposed transformer method in two metrics, which are: The Differential Sharpe Ratio (DSR) and Net Portfolio Value (NPV).

5.1.1 Quadratic Programming Method

The initial baseline model employs a multi-period portfolio optimization strategy using a quadratic programming model as demonstrated below:

$$\text{Maximize: } \lambda \cdot \sum_{t=1}^{T} h_t^\top \mu - (1-\lambda) \cdot \sum_{t=1}^{T} h_t^\top Q h_t$$

Subject to:

$$\begin{aligned}
&\mathrm{h}_{t+1} = h_t + b_t - s_t \\
&\mathrm{cash}_{t+1} = cash_t - \textstyle\sum_{i=1}^{n} b_{t,i} \cdot prices_{t,i} + \sum_{i=1}^{n} s_{t,i} \cdot prices_{t,i} \\
&\mathrm{s}_t \le h_t + b_t \\
&\mathrm{cash}_t, b_t, s_t, h_t \ge 0
\end{aligned}$$

The model is implemented with the CVXPY library and solved using the SCS solver. This approach optimizes the portfolio by balancing the risk and returns through a trade-off parameter, denoted as **λ**. The parameter **λ** acts as a risk aversion coefficient, it adjusts the emphasis between expected returns and portfolio variance. T represents the total number of day steps considered in the optimization horizon. The parameter μ represents the expected return of the assets, while h_t denotes the stock holdings at a given time t. Together, these terms constitute the cumulative expected return. Lastly, Q is the covariance matrix of asset returns, and the second term in the equation is the quadratic term, which represents the risk, also known as the variance of the portfolio.

The model is constrained to ensure both financial feasibility and risk management. Initially, the portfolio assumes no holdings and begins with a specified initial cash amount. At each time step, the stock holdings are updated based on the amounts bought and sold, ensuring consistency across time. The cash balance is also updated at each time step, accounting for the cost of buying stocks and the proceeds from selling stocks, while adhering to the portfolio's budget constraints. To prevent excessive risk, the model imposes strict non-negativity constraints on the amounts bought, sold, and held, ensuring no short-selling occurs. Additionally, stocks cannot be sold in excess of what is held in the portfolio, maintaining feasibility. The cash balance must remain non-negative at all times to prevent over-leveraging.

$$\mu = \alpha \cdot \text{recent\_returns} + (1 - \alpha) \cdot \mu$$

Furthermore, as the function above shows, the expected return μ is updated using a weighted moving average. Initially, the expected returns are calculated as the mean of the stock prices in the training dataset, providing a baseline estimate. Subsequently, daily returns are incorporated into the total expected return using a smoothing factor, α, which determines the relative weights assigned to historical data and current data. This approach ensures a balanced representation of long-term trends and recent market developments, enhancing the model's responsiveness to evolving market conditions.

The model dynamically updates stock holdings and cash balances at each time step while following constraints such as no short-selling and non-negative cash holdings. This baseline model serves as a fundamental benchmark against which more advanced models could be compared while offering insights into the benefits of employing more sophisticated reinforcement learning-based approaches.

5.1.2 Fully Connected Method

The second baseline method is actor-critic with fully connected neural networks as shown in Figure 3 [15]. This approach is computationally efficient and suitable for comparison. It provides a non-sequential structure for sequential transformer models to compare, and it has the ability to model non-linear relationships, thus providing substantial improvements in decision-making for portfolio optimization. It allows one to determine whether the added complexity of transformers or LSTMs provides a significant performance boost in terms of portfolio optimization metrics, such as risk-adjusted returns.

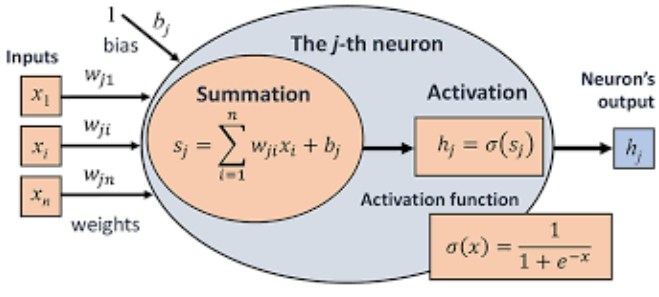

Figure 3. ANN Architecture

5.1.3 LSTM Method

The third baseline is the LSTM for both actor and critic models. LSTM is a specialized type of Recurrent Neural Network (RNN) that is designed to capture the long-term dependencies in sequential data by using the gate mechanisms [16]. The use of LSTMs for both actor and critic aims to enhance the model's ability to process complex sequential relationships inherent in financial data, thereby allowing for more informed decision-making in portfolio optimization. However, LSTMs have inherent limitations in dealing with very long sequences efficiently, as the computational complexity can create bottlenecks and lead to challenges in scalability and real-time inference. The LSTM method served as the baseline sequential model to compare with the transformer method.

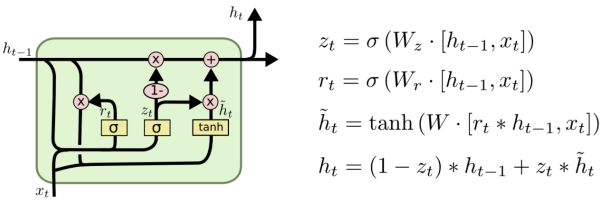

Figure 4. LSTM Gate Mechanism

5.1.4 Differential Sharpe Ratio (DSR) Metric

Inspired by work from the J.P. Morgan AI Research Group, the team adopted the Differential Sharpe Ratio (DSR) metric as the reward function, denoted as D_t [1]. This metric evaluates the effectiveness of the agent's actions on the day t within the context of portfolio management.

To provide a foundation for understanding the Differential Sharpe Ratio, it is first essential to examine the Sharpe Ratio (S_t), defined by the following function.

$$S_t = \frac{A_t}{K_t(B_t - A_t^2)^{1/2}}$$

Here, A_t is the exponential moving average (EMA) of portfolio returns up to time t, and it is expressed as:

$$A_t = \eta R_t + (1 - \eta) A_{t-1}$$

where:

R_t : The portfolio return at time t.

η: The smoothing factor ($0 < \eta < 1$), determining how much weight is given to recent returns.

Similarly, B_t denotes the EMA of the squared returns up to time t, calculated as:

$$B_t = \eta R_t^2 + (1 - \eta) B_{t-1}$$

where: R_t and η are defined as above.

As shown in the following function, DSR is the partial derivative of the S_t with respect to η. It can be represented by a combination of A_t and B_t, which takes into account the changes in average returns (ΔA_t) and changes in variance (ΔB_t), balancing their contribution to evaluate the effectiveness of the agent's actions [17].

$$D_t \equiv \frac{\partial S_t}{\partial \eta} = \frac{B_{t-1}\Delta A_t - \frac{1}{2}A_{t-1}\Delta B_t}{(B_{t-1} - A_{t-1}^2)^{3/2}}$$

This metric is chosen because it utilizes exponential moving averages and is particularly suited for the dynamic and online nature of RL environments, offering immediate, risk-adjusted performance feedback vital for RL decision-making [9].

5.1.5 Net Portfolio Value Metric (NPV)

The second metric is P_t, the Net Portfolio Value for the stock *i* at time step *t*. It is calculated by combining both C_t, the cash holding at time step *t*, and the market value of all stocks, which is a sum of the products of $p_{t,i}$, the quantity of stock *i* at time step *t*, and $v_{t,i}$, the value of stock *i* at time *t*.

$$P_t = C_t + \sum_{i=1}^{N} p_{t,i} \cdot v_{t,i}$$

This metric directly represents the final outcome of the investment policy at time step t to ensure the model's performance is also validated on a real-world objective, in contrast with the abstract DSR function.

5.2 Hyperparameter Tuning

Hyperparameter Tuning is a critical step in the development of reinforcement learning models, as it directly influences the model's ability to learn effectively and generalize well to new data. In the context of portfolio management, hyperparameters (such as the smoothing factor in the DSR reward function, the input history length, the number of heads, the block size, etc.), significantly affect the agent's decision-making quality and convergence stability. Systematic search approaches such as grid search and random search are used to identify optimal configurations that maximize the model's performance. The team evaluated each hyperparameter setting on the validation dataset for 10 epochs and picked the best setting according to the average net price (average NPV), this metric is different from NPV and only used in hyperparameter tuning.

$$\overline{P} = \frac{1}{T}\sum_{t=1}^{T}\left(C_t + \sum_{i=1}^{N} p_{t,i} \cdot v_{t,i}\right)$$

The average NPV, rather than the DSR, is chosen because, according to a study in the paper '*Deep Reinforcement Learning that Matters*' by Henderson et al., it is essential to evaluate RL agents on direct real-world metrics, rather than abstract reward functions, to ensure robustness and generalizability [18].

5.2.1 Smoothing factor η in DSR reward function

The first hyperparameter tuned by the team was the smoothing factor η in the DSR reward function. This parameter plays a crucial role in capturing a reliable signal of performance trends, particularly in the context of the inherent volatility of financial returns. By applying smoothing, η prevents the DSR from overreacting to extreme values, ensuring a more consistent and stable measure of performance over time. Furthermore, it supports generalization by mitigating the impact of outliers, while also facilitating gradual adjustments during dynamic market changes. This enhances the robustness of the evaluation process and contributes to a more effective learning process for optimizing financial strategies. After tuning, the optimal value for η in the model was determined to be 0.1.

5.2.2 Input History Length

The input history length is the length of historical data the policy model can observe to make decisions at any given point in time. As short-term patterns may be influenced by noise, recent market movements, or sentiment-driven volatility, and medium to long-term patterns may reveal momentum or reversal trends that come from broader market sentiment and macroeconomic influences, tuning this length helps ensure that the model captures the right amount of historical information needed to identify market trends and patterns and balance the short-term reactivity and long-term strategy. The best input history length is 100 trading days.

5.2.3 Number of Heads for Transformer

Transformer's multi-head attention enables the model to learn relationships from multiple sub-spaces simultaneously, which allows it to capture diverse patterns and correlations between different assets or across various time horizons. A higher number of heads will enhance the model's expressiveness but will also increase model complexity, which can lead to overfitting. The parameter is tuned in order to achieve a balance between complexity and generalizability. The best number of heads for the model is 19.

5.2.4 Block size

The block size determines the length of the input sequence considered by the self-attention mechanism, which impacts how well the model can capture short-term and long-term dependencies. Tuning this parameter allows for an optimal trade-off between model complexity and computational efficiency, which enables the model to learn the most relevant features while avoiding overfitting or underfitting. It also helps the model adapt to the unique characteristics of different financial markets and asset classes. The best block size for the model is 1024.

5.2.5 Number of block layers

The number of block layers represents the depth of the model, which directly impacts the model's ability to capture complex relationships and dependencies. Tuning this parameter will ensure the model is tailored to specific financial markets and can develop a comprehensive understanding of the financial data. The best block layer for the model is 8.

5.2.6 Learning rate and batch size

Learning rate and batch size are typical parameters that determine the model's training efficiency. The learning rate controls how quickly the model's parameters are adjusted to minimize the loss function and the batch size determines how many samples are used to estimate the gradient for each update of the model's weights. The learning rate controls the magnitude of weight updates, balancing convergence speed and training stability, while the batch size affects the quality of gradient estimates and impacts training efficiency. The best learning rate for the model is 0.005 and the best batch size is 512, while the best mini batch size is 64.

Table 1: The conclusion for parameter tuning

Hyperparameters	**Best Choice**
Smoothing factor in DSR reward	0.1
Input history length	100
Number of Heads	19
Block size	1,024
Number of block layers	8
Learning rate	0.005
Batch size/ mini-batch size for SGD	512/64

5.3 Testing Result and Discussion

The team has implemented all the proposed and baseline methods on the testing dataset given a 10000 RMB (Chinese currency) initial portfolio, with the results presented in Figures 5 and 6. Figure 5 illustrates the DSR values (reward values) over time (in terms of date numbers) for each of the tested DRL methods. A higher DSR value identifies the method has a higher incremental return meanwhile with a lower incremental risk; thus, performing better. The LSTM method achieved the highest reward value overall; however, it exhibited significant variability, with a sharp decline toward the end of the testing period. Both the Fully Connected Network and the proposed transformer method demonstrated similar performance, although the transformer method slightly outperformed the Fully Connected Network. Notably, on day 59, the transformer method achieved a substantially higher reward compared to the Fully Connected Network.

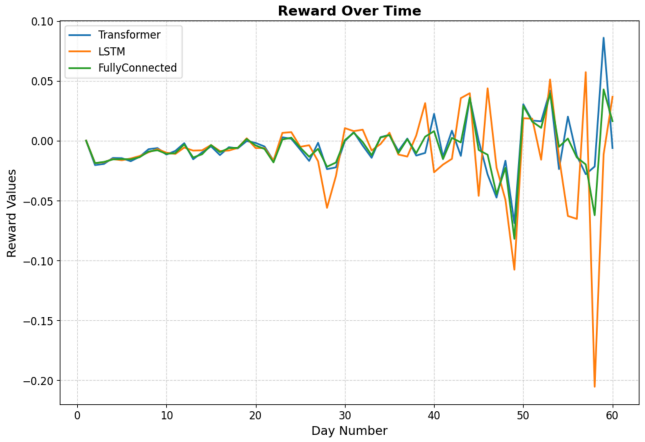

Figure 5. Comparison of DSR Value for each Method

Figure 6 shows the net portfolio value over time (based on date numbers) for each method. The quadratic optimization model demonstrated strong risk management, with low variance in portfolio value. In contrast, the LSTM method showed the same unstable behavior as observed previously. It achieved the highest return among all methods, around 10,700 on day 26, but also ended with the lowest net portfolio value at approximately 9,200. This aligns with the earlier observation, where the LSTM's low reward near the end of the testing period indicated its limited ability to manage risk. This outcome meets the team's expectations, as the LSTM's "gate" mechanism can prevent memory loss but is less effective compared to the transformer model.

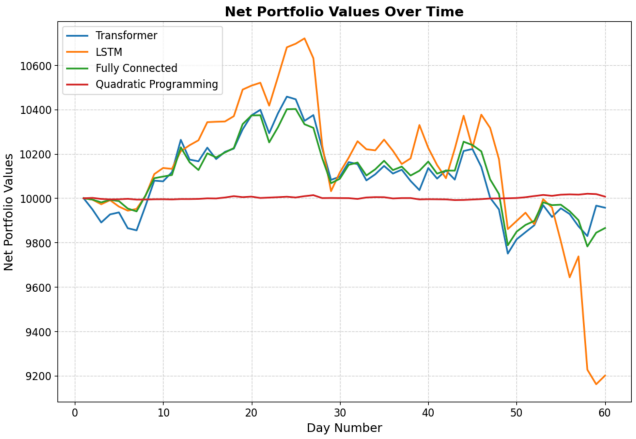

Figure 6. Comparison of the Net Portfolio Over Time

If the LSTM exhibits high variance, it would be expected that the Fully Connected Network, lacking sequence processing capabilities, would show even

greater variability. However, further analysis revealed a potential cause for unexpected behavior. As shown in Figure 7, the team computed the date (positional) encoding as described in *"Attention Is All You Need"* for the transformer model [2]. This calculation was accidentally applied to the input of the Fully Connected Network, effectively assigning sequence properties to its input. Based on this finding, the team suggests that date embeddings may also benefit traditional non-sequence-based models.

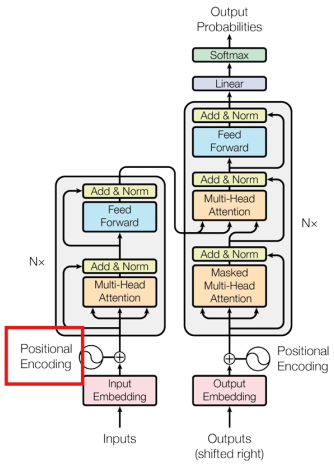

Figure 7. Date (positional) encoding in transformer

Overall, the proposed design indeed outperformed the baseline model in terms of reward functions. It implies the proposed transformer method has the best return and risk-preventing ability among DRL methods.

The proposed transformer method, while performing well, did not surpass the baseline methods to the extent anticipated. The team identifies two potential reasons for this outcome:

Insufficient Training Data: The hyperparameter-tuned transformer model used in this study is relatively large, approaching the size of a small LLM. However, unlike LLMs, which are typically trained on millions of samples, our transformer was trained on a dataset with only a few hundred samples. This small sample size limited the model's ability to capture meaningful features, significantly affecting its generalization capability. Additionally, in the context of DRL, the limited dataset size hindered the transformer model's ability to explore the environment effectively, resulting in suboptimal policy generation.

Insufficient Simulation of Environment: Although the team made every effort to construct an environment that approximates real-world trading, the simulated environment remains far from industry standards due to constraints in time and available tools. According to the study by Ndikum et al., building an industry-level financial environment that mirrors real-world market conditions requires in-depth modeling of market impact, transaction costs, and regulatory constraints [9]. If such a more complex and simulation-to-real environment could be implemented, the transformer model, with its attention mechanisms and greater complexity, would likely capture more complex features than the other methods with simpler models, thus leading to a larger performance advance.

These limitations underscore the challenges faced in fully realizing the potential of the transformer method within the current study and highlight opportunities for future improvements.

6 Conclusion and Future Work

6.1 Conclusion

From the result, the advantage of transformers over other baseline models is not that significant. This might mainly come from the limited access to stock data. Currently, the data is accessed at day level, and the sample size is relatively small for training data for transformers. If the team could access the stock data at a minute level, the sample size would be much larger. A larger sample size would enable the Transformer-based model to capture more intricate features within the data, potentially improving the model's exploratory capabilities and the quality of learned patterns.

Moreover, the current environment designed for this study is simplified compared to the complexity of real-world financial markets. This reduction in complexity limits the benefits that could be derived from employing more advanced architectures, such as Transformers or other complex actor models. Additionally, the team hasn't taken into account some indices like board economic indicators, company-specific attributes, etc.

6.2 Future Work

The team has implemented some of the future works stated below, please see the GitHub page.

Longer training dataset period:
The team would expand the period for the training dataset to include a broader range of market cycles, such as bull markets, bear markets, and periods of low volatility. The model can better learn patterns and adapt to various economic conditions by capturing data over several years. A longer training period also potentially allows the agent to develop resilience against overfitting to short-term trends and ensures its strategies are more robust when applied to unseen market scenarios.

More financial markets:
The team would choose to expand the scope by incorporating it into more financial markets, such as US stocks or cryptocurrencies. U.S. equities represent a highly liquid and well-regulated environment, while cryptocurrencies exhibit extreme volatility and operate 24/7. By training and testing the model on these additional markets, the agent can learn from a diverse set of market conditions, which offers more potential to develop a more generalized strategy that can be effective for various asset classes.

Investigate more portfolio indices:
By integrating more portfolio indices (E.g. earning growth, dividend yield, valuation yields, etc.), the model could better access the underlying values of the companies and produce higher-quality decisions. For instance, earnings growth is a critical metric for identifying companies with strong profit potential, while dividend yield offers insights into income-generating opportunities. These fundamental signals could help distinguish the outperforming and underperforming assets.

Embed more relevant external factors:
The current environment does not account for the interest earned on cash holdings and the variable transaction fees. Also, stock suspension, where trading in certain securities is halted due to regulatory or liquidity issues, is not considered in the environment. For long-term development, the team proposes integrating the external factors and document embeddings into the agent's decision-making process. The document includes embedding related news, industry regulations, and government policies into the actor's input.

These proposed future steps aim to significantly enhance the capability of the reinforcement learning model for portfolio management. By expanding to additional financial markets, integrating more diverse portfolio indices, and completing the environment to more closely reflect real-world market complexities, the model will achieve a greater level of sophistication and thus be more capable of adapting to a wide range of financial markets.

Reference

[1] S. Sood, K. Papasotiriou, M. Vaičiulis, and T. Balch, "Deep reinforcement learning for optimal portfolio allocation," icaps23, https://icaps23.icaps-conference.org/papers/finplan/FinPlan23_paper_4.pdf (accessed Dec. 16, 2024).

[2] A. Vaswani et al., "Attention is all you need," arXiv.org, https://arxiv.org/abs/1706.03762 (accessed Dec. 16, 2024).

[3] J. Moody, M. Saffell, Y. Liao, and L. Wu, "Reinforcement learning for trading systems and portfolios: Immediate vs future rewards," SpringerLink, https://link.springer.com/chapter/10.1007/978-1-4615-5625-1_10 (accessed Dec. 16, 2024).

[4] Y. Zhang et al., "Cost-sensitive portfolio selection via Deep Reinforcement Learning," arXiv.org, https://arxiv.org/abs/2003.03051 (accessed Dec. 16, 2024).

[5] L. W. Cong, K. Tang, J. Wang, and Y. Zhang, "Alphaportfolio: Direct construction through deep reinforcement learning and interpretable AI," SSRN, https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3554486 (accessed Dec. 16, 2024).

[6] E. Lezmi and J. Xu, "Time series forecasting with Transformer models and application to asset management," SSRN, https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4375798 (accessed Dec. 16, 2024).

[7] M. Gullotto, "Portfolio management and Deep Learning: Reinforcement Learning and Transformer applied to stock market data," Webthesis, https://webthesis.biblio.polito.it/20569/ (accessed Dec. 16, 2024).

[8] Y. Huang, P. Swamidass, and D. A. Raju, "The nature of innovation in Emerging Industries in China: An exploratory study - the journal of technology transfer," SpringerLink, https://link.springer.com/article/10.1007/s10961-014-9390-7?utm_source (accessed Dec. 16, 2024).

[9] P. Ndikum and S. Ndikum, "Advancing Investment Frontiers: Industry-Grade Deep Reinforcement Learning for portfolio optimization," arXiv.org, https://arxiv.org/abs/2403.07916 (accessed Dec. 16, 2024).

[10] "Reinforcement learning, Second edition," Google Books, https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf (accessed Dec. 16, 2024).

[11] N. Scheltema, "Deep reinforcement learning for recommender systems: Shaped Blog," RSS, https://www.shaped.ai/blog/deep-reinforcement-learning-for-recommender-systems--a-survey (accessed Dec. 16, 2024).

[12] "Understanding deep reinforcement learning (DRL): A comprehensive guide," What is Deep Reinforcement Learning (DRL)?, https://zilliz.com/glossary/deep-reinforcement-learning (accessed Dec. 16, 2024).

[13] Neuronlike adaptive elements that can solve difficult learning control problems | IEEE Journals & Magazine | IEEE Xplore, https://ieeexplore.ieee.org/document/6313077/ (accessed Dec. 16, 2024).

[14] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," arXiv.org, https://arxiv.org/abs/1707.06347 (accessed Dec. 16, 2024).

[15] Argatov, Ivan. "Artificial Neural Networks (Anns) as a Novel Modeling Technique in Tribology." Frontiers, www.frontiersin.org/journals/mechanical-engineering/articles/10.3389/fmech.2019.00030/full (accessed 16 Dec. 2024).

[16] "Understanding LSTM Networks." Understanding LSTM Networks -- Colah's Blog, colah.github.io/posts/2015-08-Understanding-LSTMs/ (accessed 16 Dec. 2024).

[17] AchillesJJ, "ACHILLESJJ/DSR: Differential Sharpe ratio," GitHub, https://github.com/AchillesJJ/DSR# (accessed Dec. 16, 2024).

[18] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep Reinforcement Learning that Matters," arXiv.org, https://arxiv.org/abs/1709.06560 (accessed Dec. 16, 2024).

APPENDICES

Appendix A

Table 2. All Stock Data Categories of the Dataset

Name	Type	Description
ts_code	str	Stock code
trade_date	str	Trading date
open	float	Opening price
high	float	Highest price
low	float	Lowest price
close	float	Closing price
pre_close	float	Previous closing price [Adjusted price, pre-adjusted for corporate actions]
change	float	Change in price
pct_chg	float	Percentage change [Calculated based on the adjusted previous closing price: (Today's close - Adjusted previous close) / Adjusted previous close]
vol	float	Trading volume (in lots)
amount	float	Trading amount (in thousands of yuan)
turnover_rate	float	Turnover rate (%)
turnover_rate_f	float	Turnover rate (based on free-floating shares)
volume_ratio	float	Volume ratio
pe	float	Price-to-earnings ratio (Total market cap / Net profit, empty if loss-making)
pe_ttm	float	Price-to-earnings ratio (TTM, empty if loss-making)
pb	float	Price-to-book ratio (Total market cap / Net asset value)
ps	float	Price-to-sales ratio
ps_ttm	float	Price-to-sales ratio (TTM)
dv_ratio	float	Dividend yield (%)
dv_ttm	float	Dividend yield (TTM) (%)
total_share	float	Total shares (in 10,000 shares)
float_share	float	Free-floating shares (in 10,000 shares)
free_share	float	Freely traded shares (in 10,000 shares)
total_mv	float	Total market value (in 10,000 yuan)
circ_mv	float	Circulating market value (in 10,000 yuan)