A Quantitative Portfolio Management Method with Machine Learning Optimization Algorithm

Jianghao Cui*

{*Corresponding author: cuijianghao97@163.com}

School of Computer Science, Nankai University, Tianjin, China

Abstract: In financial trading, an effective trading strategy is key to determining profit and loss. Due to the complexity and dynamics of financial markets, the automated selection of trading strategies has become the focus of modern financial research. This study utilizes deep learning and reinforcement learning methods, proposing an end-to-end deep reinforcement learning trading strategy algorithm that combines CNN and LSTM, named CLDQN. Within this framework, the CNN module is utilized to perceive dynamic market conditions of stocks and extract crucial features, while the LSTM module is responsible for learning long-term dependencies in the time series. After processing through the reinforcement learning method DQN, the algorithm makes trading decisions. To verify the effectiveness of CLDQN, we compared it with benchmark methods such as LSTM, SVM, and decision trees. Experimental results show that CLDQN's three-year cumulative return rate on four stocks is on average 1.1875%, 1.925%, and 2.3875% higher than that of LSTM, SVM, and decision trees respectively. These results not only demonstrate the superiority of the CLDQN method but also highlight its excellent scalability and robustness.

Keywords: Financial Trading Strategies; Deep Learning; Reinforcement Learning; DQN.

1 Introduction

Obtaining risk-adjusted returns on financial assets has been at the core of modern financial studies. In 1964, the Capital Asset Pricing Model (CAPM) was introduced, emphasizing that returns are primarily associated with systematic risk, with the aim of a portfolio being to eliminate non-systematic risk. This model laid the foundational theory for financial market trading. In 1970, the Efficient Market Hypothesis (EMH) was proposed, which argues that market prices fully reflect all publicly available information, thus suggesting that any technical analysis based on stock prices is ineffective. However, the behavior of actual markets does not always align with the Efficient Market Hypothesis, as demonstrated by financial phenomena such as the herd behavior and the equity premium puzzle[1].

Traditional financial trading strategies focus on price prediction, deriving from factors affecting security prices to discern market price laws[2]. However, these strategies encounter hurdles: (1) Financial time series contain noisy, unbalanced data. Manually extracting financial features to reduce noise may lack comprehensiveness. (2) Variables often show correlation and multicollinearity, skewing model parameter weight estimations. (3) Models based on historical data may not effectively predict future market shifts, limiting out-of-sample generalization. (4)

ICEMME 2023, November 17-19, Beijing, People's Republic of China
Copyright © 2024 EAI
DOI 10.4108/eai.17-11-2023.2342679

Established models can become obsolete with market changes like bull-bear transitions or momentum shifts[3].

With AI's surge, algorithms, especially deep learning (DL) and reinforcement learning (RL), are now leveraged to analyze financial data and design trading strategies[4]. DL offers end-to-end data representation, robust feature extraction, and non-linear fitting, obviating intricate internal logic needs and enhancing data handling. Its applications span algorithmic trading, risk management, fraud detection, and portfolio management. However, DL has pitfalls in stock prediction. The model's success can hinge on predictive accuracy, risking overfitting. Moreover, high predictive accuracy doesn't guarantee high returns, as models might overlook future stock trading rewards or penalties[5].

Reinforcement learning (RL) draws parallels with behavioral psychology, emphasizing exploration-exploitation balance and the transition from state to action space without needing pre-defined data[6]. Through interaction with financial markets, RL constantly optimizes strategies, aiming for maximum returns with minimal risks, resembling human cognition. This has led to its use in stock trading, portfolio allocation, bond pricing, and hedging. However, RL faces challenges like limited perception of states and constrained state-action spaces, which may not fully represent intricate markets or effectively capture time series features[7].

Deep reinforcement learning (DRL) merges deep learning's perception with RL's decision-making, excelling in complex decision-making tasks. DRL's advantages include learning from high-dimensional raw data without manual feature engineering, allowing simultaneous price prediction and investment returns[8]. It also implicitly accounts for market factors like transaction costs and liquidity. Thus, there's no need for a separate investment return model, simplifying model development. DRL's applications cover bond pricing, hedging, portfolio management, and asset-liability management.

This paper presents the CLDQN algorithm, an innovative deep reinforcement learning approach combining CNN and LSTM, to decipher financial market patterns and pinpoint optimal trading strategies[9]. The algorithm's key features are: (1) Data Pre-processing: Stock price data is converted into two-dimensional matrix images, leveraging the visual representation's efficiency in depicting complex financial information. (2) Feature Extraction: The CNN module extracts features from these matrices, highlighting short-term stock market movements and price trends. (3) Sequence Learning: The LSTM module comprehends long-term stock price patterns, deepening the model's grasp of market behaviors. (4) Decision Learning: Leveraging insights into market dynamics and patterns, the Deep Q-learning (DQN) method is used to maximize trade returns, guiding trading decisions.Empirical tests on real stock datasets show that CLDQN surpasses benchmark algorithms in performance and robustness against market unpredictability. In essence, CLDQN offers a holistic view of short-term dynamics and long-term market trends, making it a potent tool for quantitative trading.

2 Related Work

Deep learning's role in financial quantitative trading strategies can be grouped into two main categories: those emphasizing time series pattern recognition like RNN, GRU, and LSTM, and those focused on multi-source data integration and comprehensive feature extraction, such as

CNN and DNN[10]. Long Short-Term Memory (LSTM), a specific type of recurrent neural network, counters gradient vanishing and long-term dependency challenges. Its prowess in time series data analysis is evident. For instance, models leveraging LSTM often use technical indicators as inputs, with some adjusting weights based on recent performance. Some LSTMs are designed for monthly stock closing price predictions, generating smart Beta variations based on historical data. These models then produce diverse portfolio stock returns. LSTMs have been applied in forecasting stock prices, index modeling, risk evaluations, and return assessments. The Financial Sentiment Index, calculated from sentiment data across platforms, has been combined with LSTM for predicting stock returns. Other studies have fused LSTM with techniques like wavelet transform and Stacked Autoencoders (SAEs). One such method decomposes stock prices with wavelet transformation, employs SAEs for in-depth feature generation, and finally, uses LSTM for next-day closing price predictions. Methods enhancing feature extraction remain popular in financial data processing[11].

Convolutional Neural Networks (CNNs) are widely used in stock market research due to their ability to process data from multiple sources. They can extract both basic and complex features, making them suitable for tasks like price prediction and market trend analysis. A CNN-based method is introduced to predict Exchange Traded Funds (ETFs) returns. This method uses snapshots of ETFs over time, applying convolution to extract trend indicators like the Relative Strength Index (RSI), Simple Moving Average (SMA), and MACD. This data becomes the graphical input for the CNN. The model also considers correlations between markets such as the S&P 500 and NASDAQ, using both 2D and 3D CNNs. It's divided into four parts: data representation, daily feature extraction, persistent feature extraction, and forecasting. The 2D CNN focuses on individual market data, while the 3D CNN uses all market data. Predictive performance improvements ranged from 3% to 11%. Additionally, a combined CNN and LSTM framework is proposed that leverages financial news and past market data, leading to a more potent classifier. Overall, CNNs excel in financial analysis due to their consistent feature extraction capabilities across various time points, ensuring accuracy with intricate financial data.

Reinforcement learning (RL) approaches in stock market research focus on optimizing investment returns rather than direct stock price predictions. The non-differentiable nature of RL's value function allows for a more flexible design of reward functions, using multiple data sources and ensuring better model generalization. But with continuous states and actions, the action value table can become too large and complex. To overcome this, neural networks are used to approximate these tables.

In financial market analysis, deep reinforcement learning has been employed. It uses deep learning to recognize market changes and RL to make trading choices. To improve market resilience, fuzzy learning was added to handle data uncertainties, showing impressive results. Other strategies include a price-tracking approach using the Double Deep Q Network (DDQN), integrating transfer learning with Q-learning to combat overfitting, and using domain transfer to apply models across different stocks. Other solutions involve training a DQN ensemble multiple times on the same data to lower strategy risk, and combining Gated Recurrent Units (GRU) with DQN to better manage noise in time series data.

In essence, reinforcement learning-based financial trading strategies offer a promising research direction. They combine prediction, decision-making, and optimization to tackle the complexities of financial markets. Yet, they also bring about new challenges, highlighting the need for continuous research and improvement.

3 DQN Method Based on CNN and LSTM

3.1 Principles or Theory of the Model

Reinforcement learning processes are often described using Markov Decision Processes (MDP). MDP is usually defined as a model comprised of a four-tuple, which includes state, action, state transition probability distribution, and rewards. Implementing reinforcement learning algorithms usually involves the following main steps: (1) Defining the reward function. (2) Specifying the state space. (3) Designing the state transition probability distribution. (4) Seeking the optimal action policy. The description of financial reinforcement learning in this paper is as follows:

State: The state defines the financial market's environment and the information available to investors. An effective state should: 1) Incorporate key influencing factors and 2) Minimize noise. Financial market states fall into technical, fundamental, and informational categories. Technicals cover trading volume and price points, while informationals include market signals like management announcements. Fundamentals relate to a stock's intrinsic value, such as company performance and industry analysis.

Action: Action describes real-time interactions between the agent and the environment, and can be of two types: continuous and discrete. Typical actions are such as $\mathrm{a} \in \{-1,0,1\}$, where -1, 0, and 1 respectively represent selling, no action, and buying. Specifically, q_t represents the stock shares at time t, and $\mathrm{k} \in \mathrm{Z}$ (where Z represents the set of integers).

Policy: The policy, denoted as $\pi(\mathrm{s})$, maps from the state space to the action space. It describes the action a_t to be taken at probability $\mathrm{p(s, a)}$ in a certain state s, resulting in transitioning to a new state a_{t+1}.

Reward: The reward $\mathrm{r(s, a, s')}$ describes the cumulative gains from time t to T under the influence of the agent's actions. The core objective of this research is to maximize the cumulative discounted rewards over a specified period. Specifically, the learning process aims to find a policy $\pi(\mathrm{s})$ to maximize the expected cumulative discounted reward R_t, which can be formally represented as (1):

$$\max_{\pi(s)}\left[\sum_{k=0}^{\infty} \gamma^k r_{r+1+k}\right] = \max_{\pi(s)}[R_t = r_{r+1} + \gamma r_{r+2} + \gamma^2 r_{t+3} + \cdots] \tag{1}$$

Where $\gamma \in \{0,1\}$ is the discount factor, and r_t denotes the reward at time t.

For a given policy π, the expected return in a specific state s can be represented by the state value function $V^{\pi}(s)$, which is defined as (2):

$$V^{\pi}(s) = \mathbb{E}_{\pi}[\gamma^k R_t \mid s_t = s] \tag{2}$$

where γ is the discount factor, reflecting the value of future returns. In addition to state value, another core concept is the state-action value function, also commonly referred to as the Q function. The Q function evaluates the expected return of executing action a in state s and is given by (3):

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[\gamma^k R_t \mid s_t = s, \mathrm{a}_t = a] \tag{3}$$

This implies that the expected cumulative return for the agent taking action a in state s is the value of the Q function. Furthermore, we can find an optimal policy π such that for any combination of state and action, its Q function value is maximized. This optimal Q function is denoted as $Q^*(s,a)$ and satisfies (4):

$$Q^*(s,a) = \max_{\pi}\mathbb{E}[\gamma^k R_t \mid s_t = s, \mathrm{a}_t = a, \pi] \tag{4}$$

In practice, the Q-learning algorithm aims to approximate this optimal state-action value function. Moreover, the iterative objective function for the Q network is (5):

$$y = r + \gamma\max_{a'}\mathrm{Q}(s', a'; \theta_i^-) \tag{5}$$

where θ_i^- represents the parameters of the target network. During the learning process, the loss function $\mathrm{L_i}(\theta_\mathrm{i})$ is minimized iteratively, and it's defined as (6):

$$\mathrm{L_i}(\theta_\mathrm{i}) = \mathbb{E}[r + \gamma\max_{a'}\mathrm{Q}(s', a'; \theta_i^-) - \mathrm{Q}(s, a; \theta_i)]^2 \tag{6}$$

Traditional Q-learning is challenged by the continuous nature of stock prices due to its vast state space, leading to increased storage and computational needs. DQN, combining deep learning with reinforcement learning, addresses this issue. It excelled in Atari 2600 games, demonstrating its capability. Instead of a Q-table, DQN uses neural networks, optimizing storage and managing complex state spaces effectively. DQN's process includes: (1) Initialization of state, environment, replay pool, policy, and network weights. (2) Exploration and logging state-action-reward sequences. (3) Training the network using samples from the replay pool. (4) Evaluating and iterating the policy until convergence.

3.2 Model Architecture or Model Framework

The proposed quantitative trading strategy in this paper has three main components outlined in Figure 1: (1) External Environmental Dynamic Features Extraction (CNN part): This captures spatial stock market features. Historical stock price data is converted into a two-dimensional image using "candlestick charts", which depict opening, highest, lowest, and closing prices over time. Additionally, six key index candlestick charts are included, representing various stock exchange indices. (2) Time Series Pattern Analysis (LSTM part): This uses LSTM to analyze the time-series nature of stock prices, helping identify potential long-term and short-term market trends. (3) Trade Action Decision-making Part: Based on the insights from the previous sections, the model determines trading actions like buying, selling, or holding.For the input to the CNN, we generate a matrix vector based on the price fluctuation curve of 28 minutes. The row vectors represent continuous time units, i.e.$t = k+1, k+2, \ldots, k+28$ The column vectors represent the price fluctuation values after max-min normalization. As a result, we obtain a matrix of size $T_i \in \mathbb{R}^{28\times28}$, To encompass various stocks and indices, we generated seven sets of such matrix vectors, $i \in [0,7)$, which are jointly input into the CNN for feature extraction.

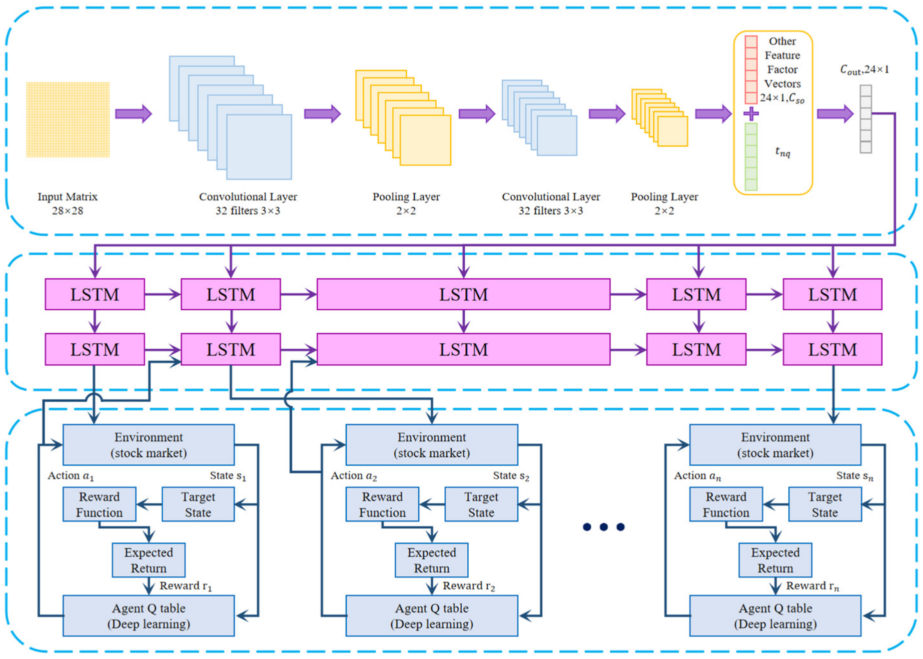


Fig. 1 CLDQN Model Architecture

This paper classifies factors influencing stock prices into four main categories based on traditional asset pricing research: (1) Liquidity Factors: Related to market trading activity. Includes turnover rate and daily trading volume. (2) Volatility Factors: Represent the unpredictability of stock prices. Encompasses metrics like 6-month and 9-month volatilities, various 30-day volatilities based on different models, MACD, MA, standard deviation of recent daily returns, highest daily return within 30 days, stochastic indicator KDJ, and price shifts over recent months. (3) Momentum Factors: Indicate the stock's trend direction. Comprises RSI and %R. (4) Fundamental Factors: Derived from a company's financial health and position. Includes metrics like the book-to-market ratio, net sales profit margin, various price ratios, dividend yield, return on assets, and various cash flow measures. To make the 24 factors comparable and dimensionally consistent, we normalized their data, producing a 24-dimensional factor vector for a holistic numerical depiction of stock traits, setting the stage for further model evaluations.

Following the CNN processing, the resultant features are fused with the other factor vector C_{so} and passed to the LSTM. The Long Short-Term Memory (LSTM) network is designed to address long-term dependencies in time-series data. It manages information flow meticulously using various gate structures, such as the forget gate, input gate, and output gate.
It's worth noting that a single layer of LSTM already offers significant capabilities, but by stacking multiple layers of LSTM, one can further amplify the depth and representational capacity of the network. As recommended by reference, this paper employs a double-layered LSTM structure, anticipating superior performance and precise time-series feature extraction through this deep setup.

In the trading environment, whenever an agent performs a trading action, it obtains the corresponding reward R_t and a new state o_t from the stock market. Concurrently, the prior hidden state h_t post its LSTM processing, serves as the neuron input for the subsequent timestep, formalized as: $h_{t+1} = LSTM(a_t, c_{out}, h_t)$. The decision action of the agent can be represented as: $a_t = \mu^{\theta}(h_t, s_t)$.
To train the network, we need to define the loss function. The definition of the loss function is (7) and (8):

$$y_t = r + \gamma Q^{\theta^-}\left(\mathrm{h}^{\mathrm{i}}_{\mathrm{t+1}}, \mu^{\theta^-}\left(\mathrm{h}^{\mathrm{i}}_{\mathrm{t+1}}, \mathrm{s}_{\mathrm{t+1}}\right)\right) \tag{7}$$

$$L(\theta) = E\left[\left(y_t - Q^{\theta}\left(\mathrm{h}^{\mathrm{i}}_{\mathrm{t+1}}, \mu^{\theta}\left(\mathrm{h}^{\mathrm{i}}_{\mathrm{t+1}}, \mathrm{s}_{\mathrm{t+1}}\right)\right)\right)^2\right] \tag{8}$$

Next, to optimize the model, we perform a partial derivative operation on the loss function, aiming to find its minimum value. In terms of strategy selection, we adopted the e-greedy policy. This strategy can balance the model's exploration and exploitation capabilities, allowing the model to choose the best action most of the time while also providing opportunities to explore new actions. To further optimize training, we also introduced an experience replay strategy. This allows for the random selection of mini-batches from historical data for training, increasing the diversity of training and preventing the model from focusing too much on data from a specific period. As a result, it better captures the overall features of the market.

4 Experiments and Results Analysis

4.1 Experimental Platform and Tools

The algorithms and models in this paper were implemented using Python 3.9, with the Tensorflow library facilitating the development of deep learning networks. Data visualization was accomplished using the Python matplotlib library version 3.5.3. The computational resources included 24 GB of RAM, a CPU of 11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz, and a GPU of RTX3060, with computational acceleration provided by CUDA 11.7. Transaction costs were set at 0.05% of the transaction amount. The mini-batch size was set to 512, with an initial learning rate of 0.01.

4.2 Experimental Setup or Experimental Configuration

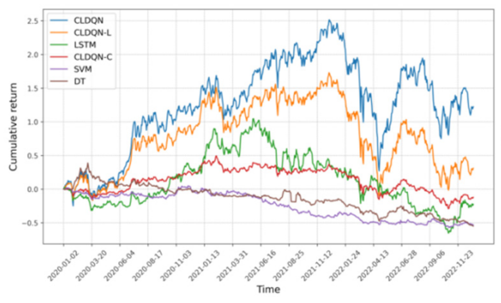
For our experiment, we sourced stock data from the Shanghai and Shenzhen markets via the Wind Terminal, focusing on four stocks: Sany Heavy Industry, Gree Electric Appliances, China Merchants Bank, and Unisplendour Corporation. Our stock selection was based on: (1) Market Value Impact: Stocks of different market values respond differently to macro market influencers. Our picks encompass high-value blue-chips (Gree Electric Appliances, China Merchants Bank), a mid-value tech growth stock (Unisplendour Corporation), and a macro-sensitive stock (Sany Heavy Industry). (2) Industry Cyclicality Variance: Stocks from diverse industries have unique cyclicality. Our chosen stocks represent a range from slow-growth, blue-chip, stable, to tech-focused, capturing industry differences. (3) Investor Attention Influence: Stock price can correlate with the amount of attention it garners. Our selected stocks showcase varying investor attention levels, letting us evaluate this relationship's impact.For data gaps, we utilized linear regression on preceding data to fill in values like opening and closing prices, and trading volume. To depict the broader market trajectory, we incorporated benchmark indices: SSE 50, CSI 300, CSI 500, and SZSE 100. These cover the spectrum from large-cap to mid-small cap market segments.

For our experiment, we used data spanning from January 2007 to December 2019 as the training set and data from January 2020 to January 2023 as the test set. We standardized transaction data post-rights adjustment to maintain comparability across various stocks.For algorithm benchmarks, we employed the LSTM model, SVM algorithm, and the decision tree. We also introduced two CLDQN variations: CLDQN-L and CLDQN-C. The former excludes the CNN element, whereas the latter bypasses the LSTM, directly feeding stock feature vectors into the LSTM. To assess the performance of these algorithms, we used four evaluation measures: cumulative return, annualized Sharpe ratio, annual return standard deviation, and maximum drawdown. Together, these metrics offer a comprehensive view of the efficacy and consistency of the quantitative trading strategies.

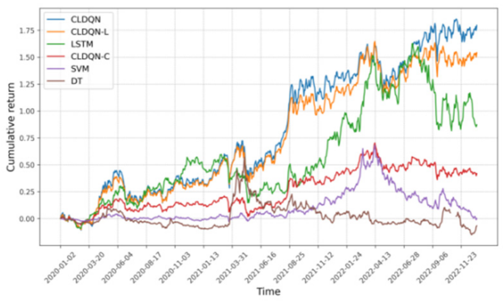
4.3 Experimental Results

4.3.1 Returns Curve for Different Stocks

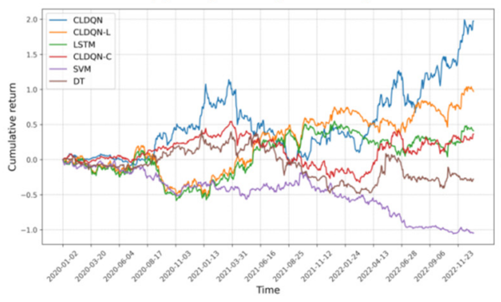
Figure 2 shows the cumulative return curves of assets for six different algorithms on data from four stocks. In the figure, the vertical axis represents the cumulative return of assets, while the horizontal axis represents the out-of-sample test period. After observation, we can draw the following conclusions:



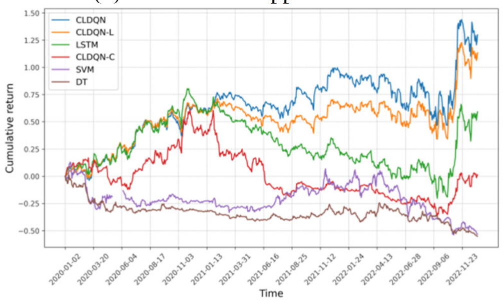
(a)Sany Heavy Industry



(b)Gree Electric Appliances



(c)China Merchants Bank



(d)Unisplendour Corporation Limited

Fig. 2 Cumulative Returns Evaluation of Stocks

(1) The yield curve of the CLDQN algorithm proposed in this study stands out amongst all the algorithms. Specifically, compared with algorithms such as LSTM, SVM, and decision trees, CLDQN performs superiorly. It's worth noting that in 2020, despite a significant decline in the benchmark algorithm, the return of the CLDQN algorithm remained relatively stable. This fully demonstrates the risk resistance and robustness of CLDQN. As can be clearly seen from the figure, the yield of the decision tree method is the lowest among all algorithms, while the performance of the LSTM algorithm lies in between. Further comparing CLDQN, CLDQN-L, and CLDQN-C, we found that when considering both time and variable interaction characteristics simultaneously, CLDQN significantly outperforms CLDQN-C. For instance, on the data of November 2022, CLDQN's prediction effect on China Merchants Bank is about 6 times that of CLDQN-C. This indicates that by effectively capturing the correlation between individual stocks and the overall market, the predictive accuracy of the CLDQN algorithm has significantly improved. Overall, these experimental results fully confirm the superiority and application potential of the CLDQN algorithm in stock quantitative trading strategies.

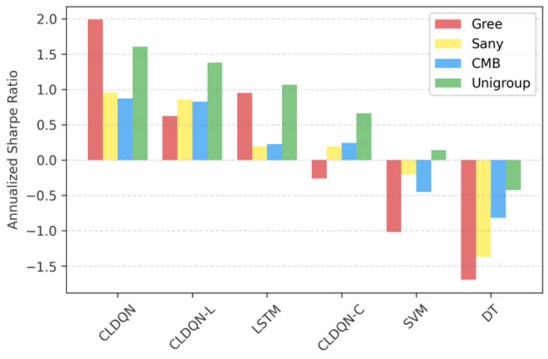
(2) Based on the experimental results, we can observe that the performance of the CLDQN-C algorithm is relatively average. This suggests that using convolutional methods to extract external environmental states has a limited contribution to enhancing algorithmic performance. Unlike image feature extraction, stock time series data will not produce two identical y-axis values at the same point in time. This uniqueness might make convolutional networks not fully adaptable to a certain extent. Furthermore, when removing LSTM from the model, the performance of the CLDQN-C algorithm saw a significant drop. This highlights the powerful capability of LSTM in capturing historical price patterns in stock time series data. Past historical data and its underlying influencing factors might reappear in future stock market dynamics. Therefore, the integration of LSTM not only discovers these hidden patterns but also effectively enhances the model's predictive and yield performance.

(3) Analysis of Momentum Reversal Effect. Based on the stock data of 2020, we analyzed the characteristics of momentum returns, especially considering the impact of the overall market environment on individual stock returns. From the results, we observed that the macro market environment has a significantly differentiated impact on different types of stocks, such as large-cap and small-cap stocks. Specifically, small-cap stocks are noticeably more sensitive to momentum reversals. For example, the momentum effect of Unisplendour Corporation significantly surpassed that of China Merchants Bank. Additionally, when the market is in a bull trend, i.e., on the rise, the momentum effect captured by the CLDQN algorithm is more pronounced than in bear markets. This emphasizes the importance of the overall market environment and stock type on the momentum effect, as well as the adaptability of CLDQN in these different scenarios.

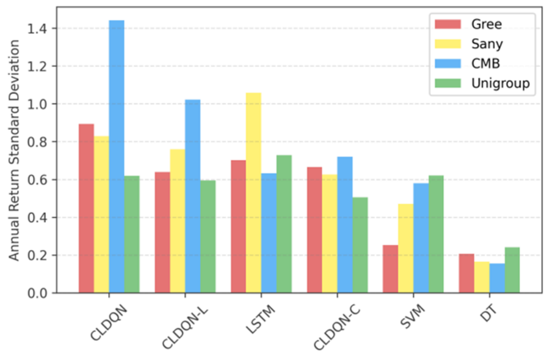
(4) Analysis of Long-term Performance Trends. From a global perspective, we noticed that the net monthly average return (i.e., monthly average return minus the market's monthly average return) showed a decreasing trend year by year between 2020 and 2023. Such a performance trend suggests that the algorithm's effectiveness in recent time periods stands out more compared to earlier periods. Given that the training dataset's time frame spans from December 2007 to December 2019, this might explain why the samples drawn from the experience replay pool tend to come from more recent periods. As the neural network parameters undergo iterative updates, the model gradually optimizes towards this time frame. However, when the model is applied to the data of 2022, it faces a market environment that is increasingly distant from the

training data. Due to the ever-changing market conditions, the model's adaptability to the new market change rules gradually weakens. This resulted in the CLDQN algorithm performing notably better in the closer temporal distance of 2020 compared to 2022.

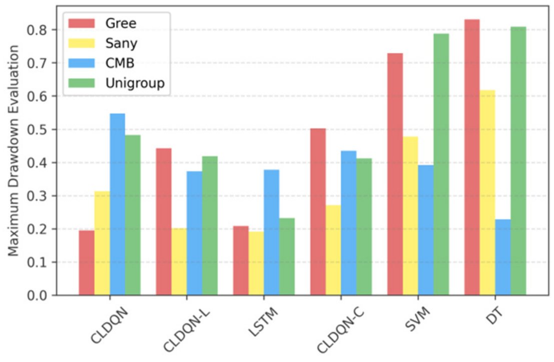
4.3.2Annualized Sharpe Ratio



(a)Annualized Sharpe Ratio Evaluation Results



(b)Annual Return Standard Deviation Evaluation Results



(c)Maximum Drawdown Rate Evaluation Results

Fig. 3: Evaluation of Annualized Sharpe Ratio, Annual Return Standard Deviation, and Maximum Drawdown Rate

Figure 3 reveals the evaluation data for various trading strategies in terms of the annualized Sharpe ratio (reflecting the compensation of asset returns to investment risk), annual return standard deviation (describing the volatility or investment risk of returns), and the maximum drawdown rate (indicating the maximum loss that might be encountered during the investment process). From the presentation in the figure, we can summarize the following observations:

(1) Robustness of the CLDQN Algorithm: The CLDQN algorithm proposed in this paper surpasses the benchmark algorithm in terms of both the annualized Sharpe ratio and the maximum drawdown rate. Notably, the annual return standard deviation of CLDQN-C is similar to that of the LSTM algorithm. This might suggest that LSTM is more sensitive to short-term temporal data changes, capable of quickly capturing and adapting to new market shifts. Meanwhile, the decision tree algorithm responds more aggressively when faced with momentum effects. Considering the characteristics of the Chinese stock market, such as relatively high market opacity and uncertainty, this means that the algorithm might face greater drawdowns after a strong initial reaction.

(2) Role of CNN in the Strategy: Based on the chart data, we observed that the influence of the CNN module on returns is not as pronounced as the LSTM module. However, the annual return standard deviation of CLDQN-C is relatively lower, suggesting that CNN can enhance the stability and robustness of the model when integrating information from multiple sources.

(3)Correlation between Market Value and Stock Characteristics: Specifically for Unisplendour , its maximum drawdown is significant, and the annualized Sharpe ratio is relatively low. This may be related to the outstanding liquidity of Unisplendour's stock, which shows a high correlation with the company's market value in cross-sectional data. In this context, the market responds more intensely to the price changes of this stock, and the company's size and book-to-market value exhibit a kind of inverse substitution relationship here.

5 Conclusion

This paper presents the CLDQN algorithm, merging CNN and LSTM for stock trading strategy through deep reinforcement learning. CLDQN utilizes vectors from price curves, stock indexes, and market factors for CNN, while leveraging LSTM for time series patterns. To bolster its robustness and deter overfitting, the algorithm adds random noise and regularization. Empirical studies indicate CLDQN's superior return rate, robustness, and scalability.

Similar to human cognitive logic, deep reinforcement learning algorithms represent a promising direction for automated trading. Future research directions can be developed from the following perspectives: (1) Behavioral Finance: Most financial RL adopts the explore-exploit pattern. Considering the volatility of the market, it's worth contemplating how to incorporate market sentiment and investor behavior as influencing factors, thereby enabling a deeper analysis through behavioral finance theories. (2) interpretability: Investors often wish to have a clear understanding of their investment logic. To address the ever-changing market conditions, research can focus on how to enhance the interpretative role of cognitive science in deep reinforcement learning portfolio optimization. (3) Introduction of Transfer Learning: Given the scarcity of valid data in real markets, it is feasible to consider applying transfer learning methods to DRL in order to generalize and adapt better to different financial market environments.

Reference

[1] Gabriele Sottocornola, Fabio Stella, Markus Zanker, and Francesco Canonaco. 2017. Towards a deep learning model for hybrid recommendation. In Proceedings of the International Conference on Web Intelligence (WI '17). Association for Computing Machinery, New York, NY, USA, 1260–1264. https://doi.org/10.1145/3106426.3110321

[2] Nishu Bansal and Satish Chandra. 2022. Solving basic and advanced human activities using LSTM. In Proceedings of the 2022 Fourteenth International Conference on Contemporary Computing (IC3-2022). Association for Computing Machinery, New York, NY, USA, 204–207. https://doi.org/10.1145/3549206.3549244

[3] M. Kaloev and G. Krastev, "Experiments Focused on Exploration in Deep Reinforcement Learning," 2021 5th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), Ankara, Turkey, 2021, pp. 351-355, doi: 10.1109/ISMSIT52890.2021.9604690.

[4] J. Kiefer and K. Dorer, "Double Deep Reinforcement Learning," 2023 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), Tomar, Portugal, 2023, pp. 17-22, doi: 10.1109/ICARSC58346.2023.10129640.

[5] L. Lyu, Y. Shen and S. Zhang, "The Advance of Reinforcement Learning and Deep Reinforcement Learning," 2022 IEEE International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA), Changchun, China, 2022, pp. 644-648, doi: 10.1109/EEBDA53927.2022.9744760.

[6] M. H. Krishna and M. M. Latha, "Complexity and Performance Evaluation of Segmented and Recursive Reinforcement Learning," 2021 IEEE 4th International Conference on Computing, Power and Communication Technologies (GUCON), Kuala Lumpur, Malaysia, 2021, pp. 1-7, doi: 10.1109/GUCON50781.2021.9573933.

[7] L. Avramelou, P. Nousi, N. Passalis, S. Doropoulos and A. Tefas, "Cryptosentiment: A Dataset and Baseline for Sentiment-Aware Deep Reinforcement Learning for Financial Trading," 2023 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW), Rhodes Island, Greece, 2023, pp. 1-5, doi: 10.1109/ICASSPW59220.2023.10193330.

[8] Y. Hu, "Digital Campus Financial Data Sharing Based on Distributed Reinforcement Learning Algorithm," 2022 International Conference on Artificial Intelligence and Autonomous Robot Systems (AIARS), Bristol, United Kingdom, 2022, pp. 51-54, doi: 10.1109/AIARS57204.2022.00019.

[9] A. Tsantekidis, N. Passalis and A. Tefas, "Improving Deep Reinforcement Learning for Financial Trading Using Neural Network Distillation," 2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP), Espoo, Finland, 2020, pp. 1-6, doi: 10.1109/MLSP49062.2020.9231849.

[10] S. Mousa, G. Ramkumar, A. J. Mohamma, B. Othman, M. S. Narayana and B. Pant, "Financial Market Sentiment Prediction Technology and Application Based on Machine Learning Model," 2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), Greater Noida, India, 2022, pp. 2279-2283, doi: 10.1109/ICACITE53722.2022.9823563.

[11] W. Wang, G. Wen and Z. Zheng, "Design of Deep Learning Mixed Language Short Text Sentiment Classification System Based on CNN Algorithm," 2022 IEEE 2nd International Conference on Mobile Networks and Wireless Communications (ICMNWC), Tumkur, Karnataka, India, 2022, pp. 1-5, doi: 10.1109/ICMNWC56175.2022.10031786.