IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 23 February 2023, accepted 15 March 2023, date of publication 20 March 2023, date of current version 4 April 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3259424

RESEARCH ARTICLE

A Novel Convolutional Neural Networks for Stock Trading Based on DDQN Algorithm

KAI CUI[1], RUIZHE HAO[2], YULING HUANG[1], JIANQING LI[1], (Senior Member, IEEE), AND YUNLIN SONG[3]

[1]School of Computer Science and Engineering, Macau University of Science and Technology, Macau, China
[2]Military Science Information Research Center, Academy of Military Science, Beijing 100142, China
[3]School of Business, Macau University of Science and Technology, Macau, China

Corresponding author: Yunlin Song (ylsong@must.edu.mo)

This work was supported in part by the Research Fund of Guangdong-Hong Kong-Macao Joint Laboratory for Intelligent Micro-Nano Optoelectronic Technology under Grant 2020B1212030010.

ABSTRACT In deep learning based stock trading strategy models, most of the research just use simple convolutional neural networks (CNN) to process stock data. But task-specific neural network structures have been proposed extensively, and their effectiveness has been demonstrated in computer vision (CV) and natural language processing (NLP) tasks. In this paper, we proposed a multi-scale convolutional neural feature extraction network (MS-CNN) for stock data, which can better extract stock trend features and thus make better decisions. The network structure inspired by the human stock trading model: in human behavior, we do not only look at a single set of stock data, but rather combine all the stock data, such as opening, closing, and trading volume, to make a comprehensive judgment. And humans will consider the current stock trend on different time scales, such as 3-Day Line and 5-Day Line. This is consistent with the two-dimensional convolution kernels commonly used in CV tasks, so we used convolution kernels of 3×3 and 5×5 in the network with two-dimensional convolution size and constructed a novel network structure for stock data. With double deep Q networks (DDQN) algorithm, we get the best performance for our network. The experimental results show that we can obtain high yield on the datasets of Dow Jones (DJI), AAPLE (AAPL), and General Electric (GE).

INDEX TERMS DRL, DDQN, CNN, stock, feature extraction network.

I. INTRODUCTION

With the development of artificial neural network research, more and more neural convolutional network (CNN) structures have been proposed. Especially for CV tasks, the network structure is being upgraded very quickly and breaking state of the art (SOTA) records in classical datasets. Feature extraction network is the most important component of the CV task and therefore have attracted a lot of research to redesign and optimize the network structure. However, in-deep learning-based stock trading networks, previous studies have mostly used simple networks for stock data processing. In the previous stock trading neural networks, the whole networks are difficult to be trained successfully. This is a limitation to the improvement of the feature extraction network complexity.

The associate editor coordinating the review of this manuscript and approving it for publication was Manuel Rosa-Zurera.

After AlphaZero beat humans based on reinforcement learning (RL), more and more scholars started to study dynamic trading strategies based on deep reinforcement learning (DRL). The process of trading can be described as decision-making process with maximizing the return while restraining the risk. One of the challenges in achieving stock trading based on DRL is to correctly analyze the state of the stock market. Most of the existing research on stock trading based on DRL analyzes the stock market through the time series features of stock data [1], such as recurrent neural network (RNN), long short-term memory (LSTM) [2], [3], [4] and gated recurrent unit (GRU). However, stock data are not only time-dependent, but also has certain spatial information, which can affect the final analysis results [5].

A. MOTIVATION

In previous DL algorithms for stock trading, the model structure used for stock data processing is simple. In DRL, when

the feature extraction network is complex, it often makes the entire model difficult to train. However, deeper neural network models can better extract high-level data feature information. These high-latitude stock characteristics can better reflect the trend information of the stock, so that the agent can make more correct decisions.

To analyze the stock market more deeply and learn the optimal dynamic trading strategies, we proposed a novel deep multi-scale feature extraction neural network used in the double deep Q networks (DDQN) algorithm framework, the entire network architecture called multi-scale double deep Q networks (MS-DDQN). MS-DDQN contains a multi-scale convolutional neural feature extraction network (MS-CNN) and DDQN to obtain the characteristic information of the stock market, which can learn and generate the optimal trading strategies. The novel MS-CNN is a policy network to automatically learn more deep information from different scales in stock data. In MS-CNN, multiple CNN layers with different kernel sizes are employed to extract the potential relationships between the different scales. In this way, we can not only obtain feature information of multiple time scales, but also extract and fuse features of various stock parameters (such as trading volume).

To test the effectiveness of the trading strategies learned by our proposed model based on feature extraction network, we compared it with other trading strategies in the Dow Jones (DJI), AAPLE (AAPL), and General Electric (GE) datasets. In the datasets of stocks with different trends, our trading strategy obtains higher profits and Sharpe ratios (SR) with better robustness. In addition, we conducted ablation experiments and the experimental results show that the learned trading strategies based on Tanh activation function outperform other activation functions, such as ELU, ReLU, and SiLU.

The main contributions of this paper are as follows.

- A novel multi-scale feature extraction network structure (MS-CNN) was proposed to analyze the stock market. We added convolution at different scales (3×3, 5×5, and 1×3) to obtain feature information at different scales. This is consistent with the human mindset of focusing on multiple temporal and spatial scales of information in the stock trading process. For example, a convolution of size 5×5 contains a 5-day line and 5 stock parameters. The convolution kernel of size 1×3 considers only one stock parameter (such us closing price) fluctuations within 3 days.
- With this multi-scale feature extraction, we observed that the agent acts on a long time scale. Specifically, the agent will determine the low point of the stock on the weekly line and buy at the low position. And the agent will not be influenced by the volatility of the daily line to take a buying action during the continuous decline of the stock.
- Experimental results on the real stock price datasets show that the trading strategy learned by MS-DDQN can obtain better profits in the stock market. Meanwhile, we also performed a series of comparison and ablation experiments to confirm the superior performance of our network structure in the selected datasets, such as using different activation functions and different number of channels.

The remainder of this paper is arranged as follows. Section II reviews the related works. Section III generally formalizes the stock trading problems. Section IV describes the architecture of MS-CNN. Section V introduces the datasets and experimental settings, then analyze the results. Section VI gives the conclusions and future works.

II. RELATED WORK

Due to advances in DL, research involving the problem of using multi-scale learning techniques to extract deep features is increasing. In multi-scale learning, models process and gather information of multiple scales [6]. In this study, the joint representation and multi-scale fusion techniques are focused. Joint representation is achieved by concatenating multiple scales with feature vectors and neural networks. Deep convolutional neural network (DCNN) models have been widely used for feature extraction in vision tasks, such as object detection, object segmentation, image classification, etc. Motivated by these applications, deep convolutional layers are commonly used as the visual feature extractor, such as AlexNet [7], VGG [8], Faster-RCNN [9], ResNet [10] and YOLO [11], [12], [13], [14]. In vision tasks, the input images are resized to a fixed size and RGB three channels. Multi-scale convolutional kernels are applied in the feature extraction network to obtain features at different scales. The feature maps extracted with different scales will enhance the representation ability of the CNN. Ronneberger et al. proposed U-Net that adopts a symmetric encoder–decoder structure to combine high-level features with low-level features to obtain richer multi-scale information [15]. Szegedy et al. proposed the Inception family CNN to learn the multi-scale features in each block by using convolutional operations with different scales for performance improvement [16], [17], [18]. Yang et al. proposed a new multi-scale convolution model based on multiple attention, introducing the attention mechanism into the structure of a Res2-block to better guide feature expression [19]. However, some studies concentrate on extracting the multi-scale information of financial time-series, such as combining short-term market features with long-term temporal features to describe time-series more precisely [20]. Cui et al. proposed a multi-scale Convolutional Neural Networks to automatically extract features at different scales and frequencies, leading to superior feature representation [21]. Thakkar et al. proposed a novel application of Pearson Correlation Coefficient (PCC) for weight initialization of Vanilla Neural Network (VNN) model to get the better initialization of weight [22]. In 2023, they proposed coefficient of variation (CV) based feature selection for stock prediction [23]. Liu et al. proposed a dual data feature

extraction method based on one single time point and multiple time points, combine short-term market features with long-term temporal features to improve the accuracy of prediction. The above methods with multi-scale information achieved remarkable improvement compared to the single-scale methods [24]. Teng et al. proposed a multi-scale local cues and hierarchical attention-based LSTM model (MLCA-LSTM) to capture the underlying price trend patterns and seek profit maximum of stock investment [25].

With the rapid development of machine learning technology, RL as one of the research directions of trading strategies [26], is employed to learn asset trading rules [27]. Théate and Ernst proposed a trading deep Q-Network (TDQN) algorithm [28]. Methods proposed by previous studies can be classified as value-based method [29], [30], [31], [32] and policy-based method [33], [34], [35], [36], [37], respectively. The value-based method uses the state value to approximate an optimal policy indirectly. On the other hand, the policy-based approach uses only the objective function to directly approximate the optimal policy. As we observed, recent research improved the techniques by merging RL method with DL. Many authors combined RL and DNN together, namely DRL to learn a good trading strategy for a given stock based on historical data. They can also be classified as value-based method [38], [39], [40], [41], [42], [43], [44], [45], [46] and policy-based method [47], [48], respectively. More recently, a few works combined multiple reinforcement learning algorithms to solve more generalized algorithmic trading problems. For instance, Zhang et al. used three RL algorithms, namely Deep Q-learning Network (DQN), Policy Gradients and Advantage Actor-Critic, to design trading strategies for continuous futures contracts and adjust trading positions based on market volatility [49]. Yang et al. proposed a novel ensemble strategy for automatically generating transaction behaviors based on Proximal Policy Optimization (PPO), Advantage Actor Critic (A2C) and Deep Deterministic Policy Gradient (DDPG) [19]. It inherited the best properties of the three algorithms to robustly adapt to different market situations for maximum returns. Li et al. proposed a robust and practical model by both DQN, asynchronous advantage actor-critic (A3C) and an LSTM for trading market [50]. And in 2022 they combined CNN and LSTM to analyze stock data and candlestick charts based on DDQN and Dueling DQN algorithms to learn optimal dynamic trading strategies [51]. Khemlichi et al. proposed the DQN algorithm which is a combination of Q-Learning and DL [52].

From our literature review, most of the previous studies on RL in finance have considered only single-scale data, and a few have further enriched the multi-scale information by sliding windows on stock. Combining multiple time series with multi-scale features, will enable agents to refer multi-dimensional information and make dynamic trading decisions based on the current market price conditions. Thus, in this study, we proposed a MS-DDQN model based on DDQN and MS-CNN to analyze the stock market more deeply and learn the optimal dynamic trading strategies.

TABLE 1. Interpretation of each actual trading operation by the market environment.

Position (POS_t)	Signal (a_t^*)	Actual Operation (a_t)	Description
0	0	0	Hold the cash.
0	1	1	Open a long position.
0	-1	-1	Open a short position.
1	0	0	Hold the long position.
1	1	0	Hold the long position.
1	-1	-1	Close the long position.
-1	0	0	Hold the short position.
-1	1	1	Close the short position.
-1	-1	0	Hold the short position.

III. PROBLEM FORMULATION

In this section, we defined the algorithmic trading sequential decision-making problem as Markov decision process (MDP) of RL.

A. STATE SPACE

A set of features that describe the current state of a stock. In general, different types of information such as historical price movements, trading volumes, financial statements, can be used as the current state. In this paper, we used the open price, highest price, lowest price, close price and trading volume from the previous $n = 20$ days as the state at time t. Two consecutive states are shown in Fig. 1.

B. ACTION SPACE

A set of actions that the agent can take. In this paper, we considered a strategy that assumes buying and selling a single security in a discrete action space in financial markets, i.e. the strategy involves phases of opening, holding and closing positions. Therefore, considering the position information POS_t, and the value of that position using an integer scalar POS_t

$$POS_t = \begin{cases} 1, & \text{if long position open,} \\ 0, & \text{if no position open,} \\ -1, & \text{if short position open.} \end{cases} \tag{1}$$

Table 1 shows interpretation of each actual trading operation by the market environment.

C. REWARD

A reward based on the agent's action in the current state. We chose risk-adjusted return, which is a composite measure that can consider both return and risk, such as the Sharpe ratio. In this paper, we chose the short-term Sharpe ratio as the reward function, which reflects information about the next K days, whereas also focusing on information about the agent's position. It can be expressed mathematically as

$$SSR_t = POS_t * SR_t, \tag{2}$$

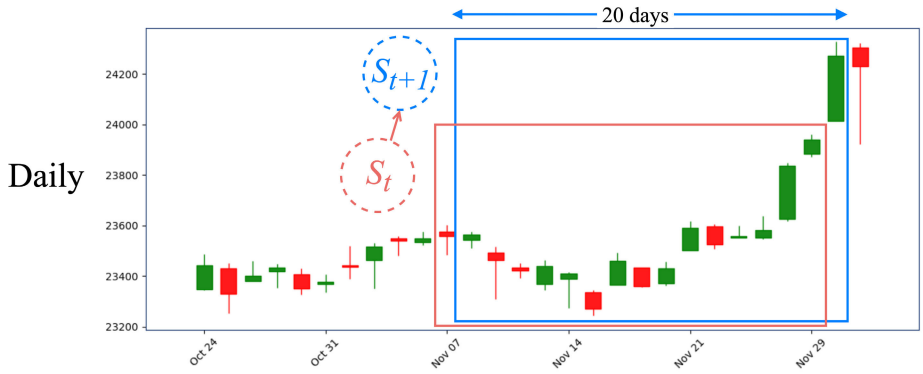


FIGURE 1. The example of state sequence.

where

$$SR_t = \frac{\text{mean}(R_t^k)}{\text{std}(R_t^k)},$$
$$R_t^k = [\frac{p_{t+1} - p_t}{p_t}, \frac{p_{t+2} - p_t}{p_t}, \cdots, \frac{p_{t+k} - p_t}{p_t}],$$

where p_t is the close price at time step t.

IV. PROPOSED METHOD

In this section, the details of the proposed trading system based on DRL are described. The frameworks of the basic models (DDQN and MS-CNN) used in this study are described. Although DDQN is useful in DL, it does not perform well in the feature learning. Thus, a MS-CNN structure was proposed to solve the aforementioned problem.

A. SYSTEM OVERVIEW

In this section, a brief overview of our trading system based on DRL is provided, including the proposed novel network structure based on DCNN. Our system is divided into two parts. In the first part, data from the raw daily prices and trading volume are processed. Raw daily data including open price, highest price, lowest price, close price, and trading volume are processed by deep multi-scale neural network. In the second part, the RL algorithm is executed. The agent observes the state, then executes the action resulting from its policy and receives corresponding rewards. Finally the agent will generate the optimal trading strategy by interacting with the environment in the RL framework. A good trading strategy is not just maximizing profits on a single trade, but also making long-term profits. The structure of the multi-scale reinforcement trading system is shown in Fig. 2.

As presented in Fig. 2, for episode $t = \{1, 2, \cdots, T\}$, agent will generate a sequence of transactions with a reward of $\{r_1, r_2, \cdots, r_t, \cdots, r_T\}$. Meanwhile, the discounted return G_t, representing the expected return of agent, is defined as

$$G_t = r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + \cdots + \gamma^{T-t} \cdot r_T, \quad (3)$$

where the parameter γ is the discount factor ($\gamma \in [0, 1]$).

Therefore, we estimated the expected value of $G_t(s_t)$ by observing the trading, defined as

$$\begin{aligned} Q_\pi(s, a) &= \mathbb{E}[G_t | S_t = s, A_t = a] \\ &= \mathbb{E}[r_t + \gamma \cdot (r_{t+1} + \gamma r_{t+2} + \cdots) | S_t = s, A_t = a] \\ &= \mathbb{E}[r_t + \gamma \cdot G_{t+1} | S_t = s, A_t = a] \\ &= \mathbb{E}[r_t + \gamma \cdot Q_\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a]. \quad (4) \end{aligned}$$

$Q(s, a)$ is a metric to evaluate how good a specific state, action pair is relative to others. It is the expected return for taking action a in the current state s. To update the current state-action value function, we adopted the next state-action action value to estimate it. Here, we adopted an off-policy value-based method, Q learning to obtain the optimal state-action value function [53]. Therefore, the updated equation of the state-action value can be defined as

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) - \alpha[r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]. \quad (5)$$

Q learning finds an optimal policy that maximizes the expected value of the total reward over all successive steps, starting from the current state. The goal of Q learning is to learn an optimal policy that guides the agent to take the correct action in a certain situation. However, the state of the stock market is numerous so that we transfer the problem of updating the Q-value tabular to the problem of function approximation. A function approximator is needed to represent each state and map each state to an action, buy, hold or sell.

DQN is one of the RL algorithms, which is an off-policy value-based approach. DQN solves the problem of unstable convergence with approximate learning state-action value function method by combining Q learning and DL with two key technologies of experience replay and target network. Experience replay is a method to reduce the correlation between training data, by randomly batch sampling between N data points and training experience. The algorithm stores the latest N data points in an in-memory buffer and randomly selects a batch-sized number of data points to fit the model at each step/iteration.

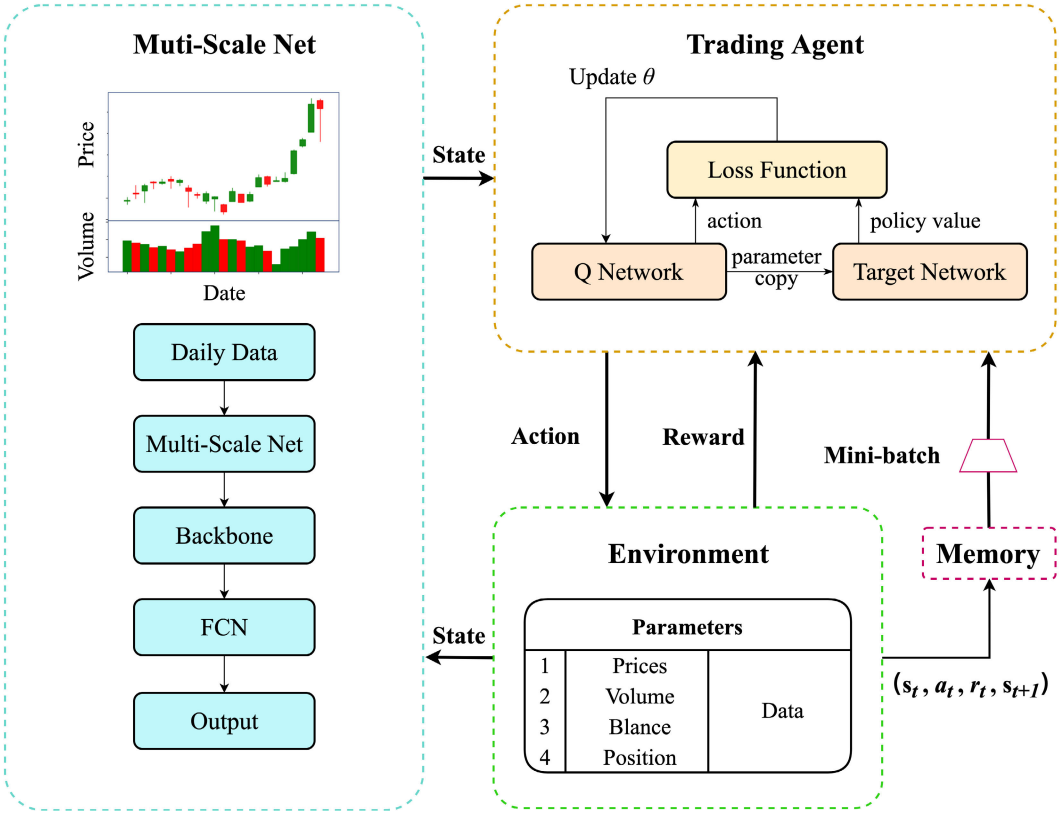


FIGURE 2. Structure of the multiscale reinforcement trading system.

B. MULTI-SCALE DEEP CONVOLUTIONAL NEURAL NETWORK

In this study, the MS-DDQN framework is presented, depicted schematically in Fig. 3. The MS-DDQN learns from the volatility of daily stock prices and trading volumes.

1) MULTI-SCALE NET

The main purpose of this network is to extract multi-scale features from stock data and fuse them into a feature map. Similar to the structure of neural networks in CV tasks, the MS-CNN contains three components. Analyzing the human stock trading mindset: (1)People will focus on single day stock data in their trading and analyze these data individually on a time scale. (2)People will perform stock trend analysis on a fixed time scale, with the 3-day and 5-day lines being important reference indicators.

Based on the results of our analysis, the multi-scale net contains three different modules: (1)The single-scale feature extraction module, which uses a one-dimensional convolution to extract time-domain features for 5 parameters and then stacks them to 5-dimensional features; (2)The 3×3 extraction module, which regards the data as a single-channel 2D image and uses a 2D convolution with a convolution kernel size of 3×3. This convolution mimics the behavior of a human observing a three-day line; (3)The 5×5 module, which also regards the data as a single-channel image and uses a convolution kernel size of 5×5 for global feature extraction. This convolution mimics the behavior of a human observing a three-day line. Finally, features from three modules is stacked as an 8-channel 2D feature layer as the output of the multi-scale net.

2) BACKBONE

We used a deep convolutional layer as the backbone network for feature extraction. In backbone, we all used traditional 3×3 convolution, added batch normalization (BN) layer and activation function to the network. Considering that the network is prone to overfitting in DL, especially after using deep convolution. Therefore, we added a pooling layer to the network. With the experience in CV networks, we added attention modules to the network as well and achieved even better performance.

3) FULLY CONNECTED NET

After obtaining the feature maps, we finally transformed the feature map into the output of MS-CNN. For this purpose, we used two fully connected layers to transform the network features into three parameters.

This network is tasked with computing $Q(s, a; \theta)$, where θ is the set of weights and biases of the network, an approximation of the action value function for the market environment. It is well known that approximating the action-value function Q by the function is advantageous, especially when the state space is large or continuous. Giving the current state s,

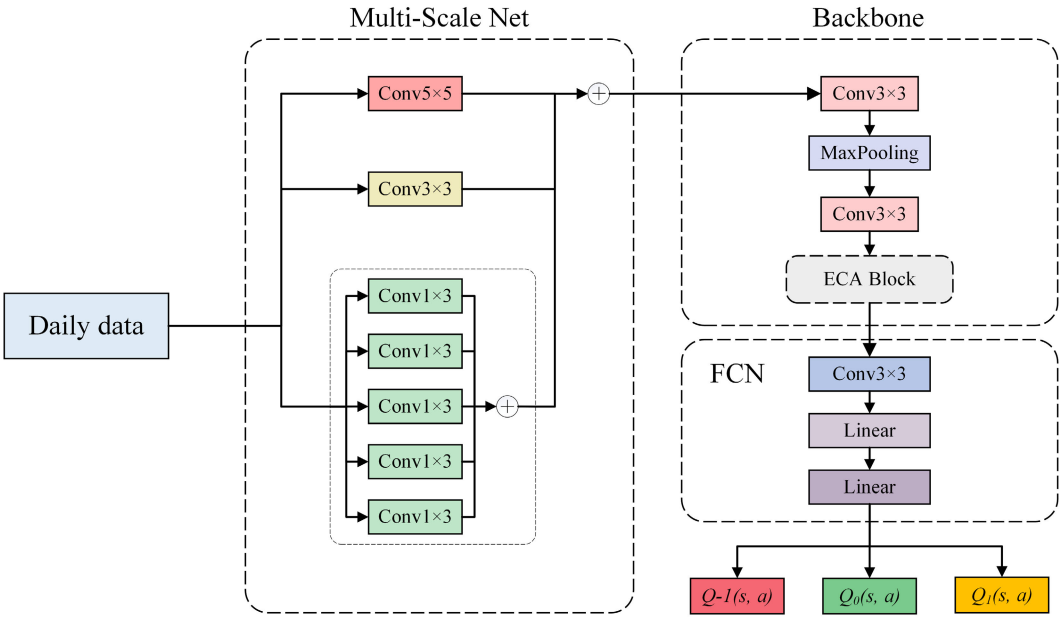


FIGURE 3. Multi-scale feature extraction network structure diagram.

selecting the action a by the ϵ-greedy method, observing the next state s' and reward r_t, then we could get the current state-action value as $Q(s, a)$. Therefore, the current optimal state action can be estimated as

$$Q^*(s, a) = \mathbb{E}[r_t + \gamma \max_{A_{t+1}} Q(S_{t+1}, A_{t+1}) | S_t = s, A_t = a], \tag{6}$$

where A_{t+1} is the next action selected by ϵ-greedy.

However, in the above equation, the max operator uses the same parameters both to select and to evaluate an action. This makes it more likely that an overestimated value will be chosen, leading to an overly optimistic estimate of value.

C. MS-DDQN LEARNING

In this paper, we applied DDQN algorithm to train the agent and get the optimal policy. In DDQN, the network is split into two separate sub-networks: an actor network and a target network. The actor network selects the actions taken by the agent and the target network generates the Q value for the action. They are identical in structure, and the weights are copied from the actor network to the target network after the regular number of steps, so that their weights are synchronized. Therefore, we used DDQN algorithm to generate optimal trading strategies by interacting with the environment. In the process of stock trading, if the agent chooses the trading action to be profitable, it will get a positive return. Conversely, if a loss occurs after choosing a trading action, the agent will receive a negative reward. These rewards motivate the agent to take the correct action in future action choices. DDQN changes the calculation of the Q value of the target network and solves the Q value overestimation problem in the DQN algorithm. The formula for calculating the Q value of the target network in the DDQN algorithm is

$$y_t = r_t + \gamma Q(s_{t+1}, \text{argmin}_a(s_{t+1}, a; \theta); \theta^-), \tag{7}$$

where θ and θ^- represent the parameters in the main network and target network, respectively.

Herein, we analyzed the stock market, from stock daily prices and trading volumes to obtain stock market state features representation, to help the agent learn the optimal dynamic trading strategy. The current state s_t is processed by MS-CNN to get the current state-action value Q, which is a 3-dimension vector. Then the current action a is selected with ϵ-greedy algorithm. The reward r and the next state s_{t+1} is observed. Therefore, the process of training the model is displayed in algorithm 1.

V. EXPERIMENT AND RESULTS

In this part, we compare the proposed MS-DDQN algorithm with other algorithms by implementing experiments on U.S. Stock Market based on PyTorch to illustrate its effectiveness and superiority.

A. DATASETS

The proposed method was tested in DJI, AAPL, and GE by collecting the daily market data from the Yahoo Finance. At the same time, we chose the past 15 years as the trading period, which is divided into the training set from Jan-01-2007 to Dec-31-2017 and the test set from Jan-01-2018 to Dec-31-2020. The price history of each asset is shown in Fig. 4 to 6.

Algorithm 1 MS-DDQN Algorithm

Input: the asset of open price, highest price, lowest price, close price, trading volume;

Initialize optimal replay memory D to capacity N;
Initialize the policy DCNN with random weights θ;
Initialize target DCNN with weights $\theta = \theta^-$;
for episode = 1 to N **do**
    Initialize sequence $s_1 = \{x_1\}$ and preprocessed state $\phi_1 = \phi(s_1)$;
    for t=1 to T **do**
        Select a_t with ϵ-greedy method;
        Otherwise, select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$;
        Execute the action a_t in the environment and get reward r_t and observe next state s_{t+1}, $\phi_{t+1} = \phi(s_{t+1})$;
        Store the transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D;
        Sample random mini-batch of transitions $((\phi_j, a_j, r_j, \phi_{j+1})$ from D;
        Set:

$$y_i = \begin{cases} r_i, & \text{if episode terminates at step } i+1, \\ r_i + \gamma Q^-(\phi_{i+1}, a'; \theta^-), & \text{otherwise.} \end{cases}$$

        Train the network with loss function $L(\theta) = E[(y_i - Q(\phi_i, a_i; \theta))^2]$;
        Set $s_t \leftarrow s_{t+1}$;
        Update the target DCNN parameters $\theta = \theta^-$ every C steps;
    end for
end for

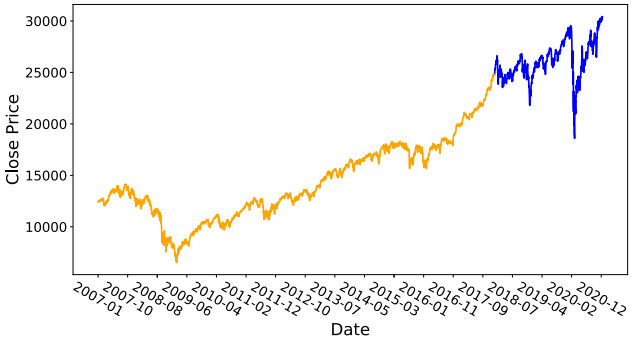


FIGURE 4. The DJI price movement of assets[01/01/2007 - 12/31/2020] [Train set in orange, Test set in blue].

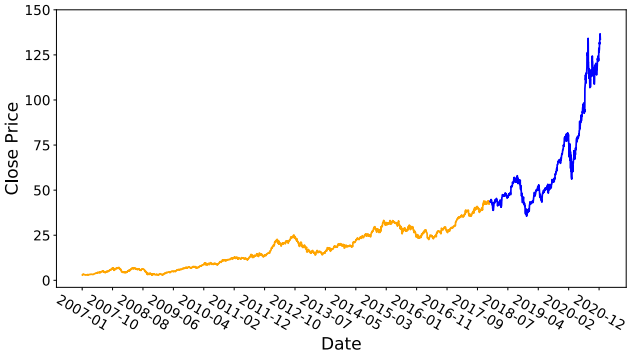


FIGURE 5. The AAPL price movement of assets[01/01/2007 - 12/31/2020] [Train set in orange, Test set in blue].

B. EVALUATION METRICS

To comprehensively and objectively assess the performance of a trading strategy, we selected three performance metrics shown below.

- **Profit:** Profit quantifies the capital gained or lost in trading activities. At each time point t, we use (8) to calculate profit for each time step t, where the current wealth is W_t and initial investment is W_0.

$$Profit_t = W_t - W_0 \tag{8}$$

- **Sharpe ratio:** It displays the average return earned more than the risk-free rate per unit total risk and is computed in (9), where R_f is the return of the risk-free asset, and $E\{R_p\}$ is the expected value of the portfolio value. In this study, we assumed $R_f = 0$.

$$SR = \frac{E\{R_p\} - R_f}{\sigma_p} \tag{9}$$

- **Annualised return(AR):** AR is the annual average profit and loss (in %) provided by an investment during trading activity. AR is defined as

$$\text{AR} = ((1 + \text{Cumulative Return})^{\frac{365}{\text{Days Held}}} - 1) * 100 \tag{10}$$

C. BASELINE METHODS

To objectively compared the performance of our proposed method, we adopt some as strategies as baselines.

- **Buy and Hold (B&H):** In this strategy, an investor selects an asset and takes a long position in the first step of investing. Assets purchased are held until the end of the period, regardless of price fluctuations.
- **Sell and Hold (S&H):** In this strategy, an investor selects an asset and takes a short position in the first step of investing. Assets purchased are held until the end of the period, regardless of price fluctuations.

FIGURE 6. The GE price movement of assets[01/01/2007 - 12/31/2020] [Train set in orange, Test set in blue].

- **DQN-pattern:** Proposed by Taghian et al. [46] learning fitted trading rules based on the candlestick chart patterns for a given stock. These patterns are extracted from a binary vector that represents the appearance of each introduced popular candlestick pattern, including the Opening, High, Low, and Close data.
- **DQN-vanilla:** Proposed by Taghian et al. [46] learning fitted trading rules based on the raw Opening, High, Low, and Close prices representation of candles.
- **TDQN:** Proposed by Théate and Ernst [28]. We downloaded the source code from github and took the result with its highest AR a t the same time period.
- **Stock trading strategies based on deep reinforcement learning:** Proposed by Li et al. [51]. We directly took out the experimental results (AR) as baseline.

D. EXPERIMENTAL SETUPS

We compared proposed methods with six baselines. The initial capital was set at $500,000, the transaction cost was 0.01%. For the DL part, the daily window length M were set to 20.

To ensure that the inputs are all within a reasonable range, we normalized the data by calculating the mean and variance, used Adam to optimize all models with a learning rate of 0.001. The models were implemented using the Pytorch library in Python, and the training stopped after 100 iterations. As for the DQN-pattern and DQN-vanilla method, the code provided by [46] was used to build the system, the initial capital was also set at $500,000.

E. EXPERIMENTAL RESULTS

1) PERFORMANCE ANALYSIS

The results are shown in Fig. 7 to 9. The initial cash was $500,000. We can observe that our MS-DDQN model outperforms other baseline models. No matter how the price fluctuates, these purple curves keep rising and rarely drop sharply in all figures. It means that the MS-DDQN algorithm has a more robust performance and performs better than other baseline algorithms in a downward market. Moreover, it adapts to new trends very quickly, which means it significantly captures the new market opportunity to make profits when the trend suddenly goes to the opposite. The next observation

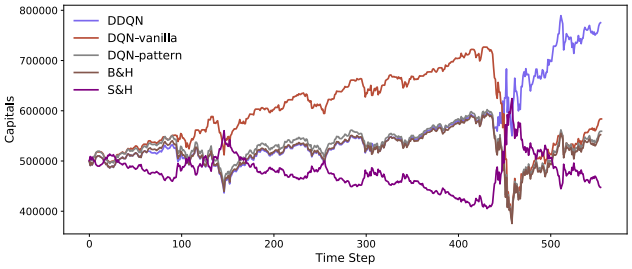


FIGURE 7. Comparison of capital return by different methods on DJI test set.

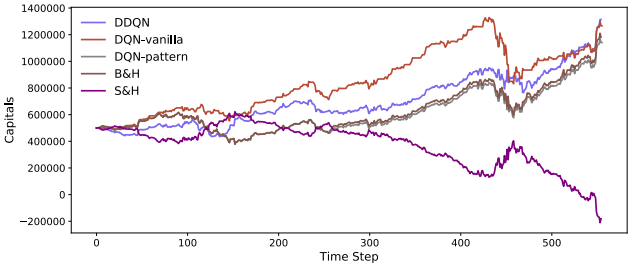


FIGURE 8. Comparison of capital return by different methods on AAPL test set.

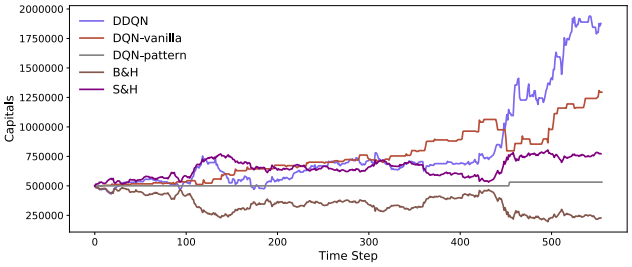


FIGURE 9. Comparison of capital return by different methods on GE test set.

is that DQN-vanilla performs nearly well as MS-DDQN in the AAPL dataset. However, DQN-vanilla is sensitive to the market volatility.

Furthermore, table 2 gives the total profits, SR and AR of different trading systems on DJI, AAPL, and GE datasets. MS-DDQN beats all baseline models in all tests. MS-DDQN outperforms DQN-vanilla 16.94% in the DJI dataset, 2.33% in the AAPL dataset period and 24.11% in the GE dataset on annualised return. DQN-vanilla only slightly exceed MS-DDQN in the GE dataset on SR. And take account of total profits and annualised return in the table, MS-DDQN behaves more positively than DQN-vanilla. Besides, the MS-DDQN algorithm can get more profit than other baselines when the price goes down. Especially, the sharpe ratio can reach 1.831, and the annualized return is 82.95%. Compared to TDQN, we achieve higher performance on the DJI and AAPL datasets. In addition, the AR of Ref [51] is 55.71% on APPL, and our AR is also 7% higher than them. These results further explain MS-DDQN algorithm is more effective.

2) TRADING SIGNAL ANALYSIS

In this section, we show the trading points of MS-DDQN algorithms, as shown in Fig. 10 to 12. We observed that MS-DDQN takes actions nearly at bottom or peak, it indeed

TABLE 2. Performances of different trading methods on different assets.

Assets	Metrics	B&H	S&H	DQN-pattern	DQN-vanilla	TDQN [28]	DCNNQN
DJI	Profit	53,608	-53,608	59,233	83,737	152,220	**276,461**
	SR	0.304	-0.079	0.325	0.414	0.498	**0.928**
	AR	11.34	-2.88	12.16	13.97	16.39	**30.91**
AAPL	Profit	679,847	-679,847	640,594	766,268	640,005	**812,744**
	SR	1.271	-0.234	1.230	1.563	1.345	**1.563**
	AR	57.80	-100.00	55.95	59.22	54.17	**61.55**
GE	Profit	-273,017	273,017	31,425	794,302	-	**1,376,148**
	SR	-0.425	0.918	0.673	**1.831**	-	1.678
	AR	-35.55	30.71	4.09	58.84	-	**82.95**

[1] **SR**: Sharpe ratio; **AR**: Annualised return.

TABLE 3. Performance analysis of DDQN with different activation functions.

Assets	Metrics	Tanh	ELU	ReLU	SiLU
DJI	Profit	**276,461**	167,590	181,329	244,307
	SR	**0.928**	0.653	0.692	0.844
	AR	**30.91**	22.07	23.29	28.73
AAPL	Profit	**812,744**	622,615	795,997	481,326
	SR	**1.563**	1.327	1.430	1.103
	AR	61.55	53.45	**62.11**	46.81
GE	Profit	**1,376,148**	1,221,228	89,831	1,369,053
	SR	**1.678**	1.585	0.385	1.590
	AR	82.95	78.71	20.81	**84.03**
Mean	Profit	**821,784.33**	670,477.67	213,431.40	698,228.67
	SR	**1.390**	1.188	0.627	1.179
	AR	**58.47**	51.41	26.55	53.19

[1] **SR**: Sharpe ratio; **AR**: Annualised return.

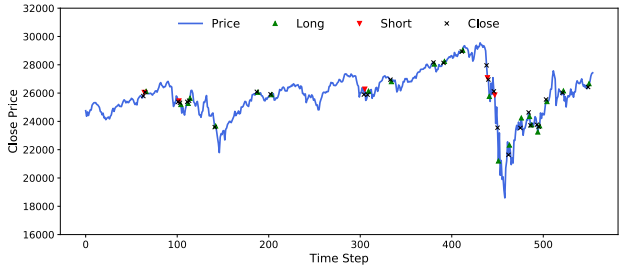


FIGURE 10. The trading signals on DJI.

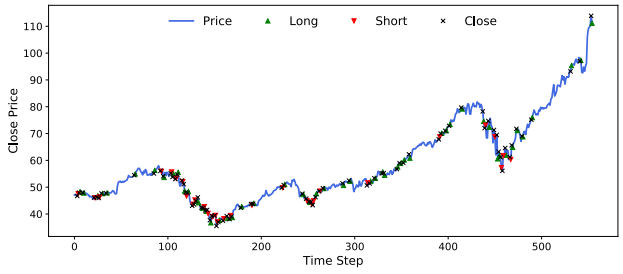


FIGURE 11. The trading signals on AAPL.

learned patterns and makes valuable decisions. In real human trading, it is easy to obtain profit in a bull market, but it is difficult to obtain stable profit or stop loss in a bear or consolidation market. MS-DDQN performs well whether in the downward movement (the GE stock), and a consolidation following a upward movement (the AAPL stock). According to these results, we can conclude that the proposed MS-DDQN algorithm can obtain stable profits in different market states.

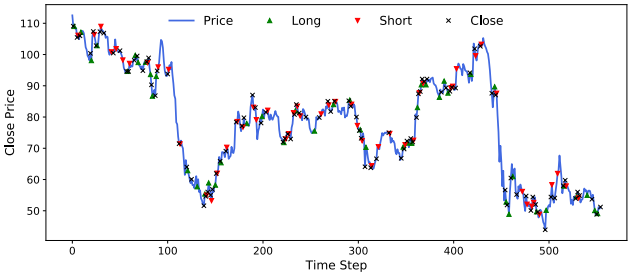


FIGURE 12. The trading signals on GE.

3) COMPARISON EXPERIMENTS

In this section, we perform several comparison experiments to explain the importance of the activation function. Activation functions are an important part of machine learning. Herein, we compared ELU, ReLU, and SiLU activation functions with Tanh in MS-DDQN algorithm. The performance metrics are shown in table 3 and it is concluded that Tanh performs well in the proposed model. The experimental results show that the network model with the selected Tanh activation function can obtain higher gains.

4) OTHER EXPERIMENTS

A series of experiments were also conducted to select the best network structure.

- **Pooling layer:** The training was performed using maximum pooling and average pooling, and the results show that the maximum pooling layer can obtain higher and

more stable returns. In addition, we placed the pooling layer at different locations in the network. The results show that the best results are obtained when the pooling layer is set in the middle of the backbone.

- **BN layer:** BN layer is a commonly used structure in CV networks, so we conducted a series of experiments for BN layer. The experimental results show that our model can be trained stably only when the BN layer is in the position after activation function. When it is in the position before the activation function or discarded, it can lead to the problem of gradient disappearance.
- **Number of channels:** The number of channels in the network often determines the performance of the model, so we varied the number of channels for comparison experiments. The experimental results show that the number of channels used in our network is optimal: when increasing the number of channels, the network becomes unstable and difficult to be trained; when decreasing the number of channels, the performance of the network decreases significantly.

VI. CONCLUSION

This paper applied DRL to financial trading problems. MS-DDQN updates parameters using experience replay algorithm and two network during training. The selection and evaluation of the action are separated, using one set of weights for determining the greedy policy and another set for determining its value. To perform better feature extraction, we proposed a multi-scale feature extraction network. This multi-scale network is inspired by CV tasks and therefore we used experience to building the network structure, such as the structure of DCNN and using pooling layer. Finally we demonstrated the effectiveness of our model by comparing it with B&H, S&H, DQN-pattern, DQN-vanilla, TDQN, and another strategy on the datasets of DJI, AAPL, and GE.

Though MS-DDQN does really better compared with other trading systems, there are many aspects to explore and improve. On the one hand, we only used the raw daily prices and trading volume of quotes for trading, but there are many other data for trading, such as fundamental data and public opinion data. We could properly integrate them together to achieve better performance. On the other hand, we can consider introducing multi-scale data into our algorithm, such as weekly, hourly, minute data.

REFERENCES

[1] K. Chaudhari and A. Thakkar, "Data fusion with factored quantization for stock trend prediction using neural networks," *Inf. Process. Manage.*, vol. 60, no. 3, May 2023, Art. no. 103293.

[2] A. Thakkar and K. Chaudhari, "CREST: Cross-reference to exchange-based stock trend prediction using long short-term memory," *Proc. Comput. Sci.*, vol. 167, pp. 616–625, Jan. 2020, doi: 10.1016/j.procs.2020.03.328.

[3] K. Chaudhari and A. Thakkar, "ICREST: International cross-reference to exchange-based stock trend prediction using long short-term memory," in *Applied Soft Computing and Communication Networks* (Lecture Notes in Networks and Systems), S. M. Thampi, J. L. Mauri, X. Fernando, R. Boppana, S. Geetha, A. Sikora, Eds. Singapore: Springer, 2021, pp. 323–338, doi: 10.1007/978-981-33-6173-7_22.

[4] A. Thakkar and K. Chaudhari, "Information fusion-based genetic algorithm with long short-term memory for stock price and trend prediction," *Appl. Soft Comput.*, vol. 128, Oct. 2022, Art. no. 109428, doi: 10.1016/j.asoc.2022.109428.

[5] W. Khan, M. A. Ghazanfar, M. A. Azam, A. Karami, K. H. Alyoubi, and A. S. Alfakeeh, "Stock market prediction using machine learning classifiers and social media, news," *J. Ambient Intell. Humanized Comput.*, vol. 13, no. 7, pp. 3433–3456, Jul. 2022.

[6] Y. Wang, S. W. Cheung, E. T. Chung, Y. Efendiev, and M. Wang, "Deep multiscale model learning," *J. Comput. Phys.*, vol. 406, Apr. 2020, Art. no. 109071, doi: 10.1016/j.jcp.2019.109071.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," 2015, *arXiv:1506.01497*.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015, *arXiv:1512.03385*.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2015, *arXiv:1506.02640*.

[12] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," 2016, *arXiv:1612.08242*.

[13] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*.

[14] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," 2020, *arXiv:2004.10934*.

[15] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention MICCAI 2015* (Lecture Notes in Computer Science), N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham, Switzerland: Springer, 2015, pp. 234–241, doi: 10.1007/978-3-319-24574-4_28.

[16] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2014, *arXiv:1409.4842*.

[17] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," 2015, *arXiv:1512.00567*.

[18] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, Inception-ResNet and the impact of residual connections on learning," 2016, *arXiv:1602.07261*.

[19] Y. Yang, C. Xu, F. Dong, and X. Wang, "A new multi-scale convolutional model based on multiple attention for image classification," *Appl. Sci.*, vol. 10, no. 1, p. 101, Dec. 2019, doi: 10.3390/app10010101.

[20] M. M. Dacorogna, C. L. Gauvreau, U. A. Müller, R. B. Olsen, and O. V. Pictet, "Changing time scale for short-term forecasting in financial markets," *J. Forecasting*, vol. 15, no. 3, pp. 203–227, 1996, doi: 10.1002/(SICI)1099-131X(199604)15:3<203::AID-FOR619>3.0.CO;2-Y.

[21] Z. Cui, W. Chen, and Y. Chen, "Multi-scale convolutional neural networks for time series classification," 2016, *arXiv:1603.06995*.

[22] A. Thakkar, D. Patel, and P. Shah, "Pearson correlation coefficient-based performance enhancement of vanilla neural network for stock trend prediction," *Neural Comput. Appl.*, vol. 33, no. 24, pp. 16985–17000, Dec. 2021, doi: 10.1007/s00521-021-06290-2.

[23] K. Chaudhari and A. Thakkar, "Neural network systems with an integrated coefficient of variation-based feature selection for stock price and trend prediction," *Expert Syst. Appl.*, vol. 219, Jun. 2023, Art. no. 119527.

[24] G. Liu, Y. Mao, Q. Sun, H. Huang, W. Gao, X. Li, J. Shen, R. Li, and X. Wang, "Multi-scale two-way deep neural network for stock trend prediction," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Jul. 2020, pp. 4555–4561, doi: 10.24963/ijcai.2020/628.

[25] X. Teng, X. Zhang, and Z. Luo, "Multi-scale local cues and hierarchical attention-based LSTM for stock price trend prediction," *Neurocomputing*, vol. 505, pp. 92–100, Sep. 2022, doi: 10.1016/j.neucom.2022.07.016.

[26] A. Mosavi, Y. Faghan, P. Ghamisi, P. Duan, S. F. Ardabili, E. Salwana, and S. S. Band, "Comprehensive review of deep reinforcement learning methods and applications in economics," *Mathematics*, vol. 8, no. 10, p. 1640, Sep. 2020, doi: 10.3390/math8101640.

[27] A. Thakkar and K. Chaudhari, "A comprehensive survey on deep neural networks for stock market: The need, challenges, and future directions," *Expert Syst. Appl.*, vol. 177, Sep. 2021, Art. no. 114800, doi: 10.1016/j.eswa.2021.114800.

[28] T. Théate and D. Ernst, "An application of deep reinforcement learning to algorithmic trading," *Expert Syst. Appl.*, vol. 173, Jul. 2021, Art. no. 114632, doi: 10.1016/j.eswa.2021.114632.
[29] G. Tesauro, "TD-gammon, a self-teaching backgammon program, achieves master-level play," *Neural Comput.*, vol. 6, no. 2, pp. 215–219, Mar. 1994, doi: 10.1162/neco.1994.6.2.215.
[30] R. Neuneier, "Enhancing Q-learning for optimal asset allocation," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 10, M. I. Jordan, M. J. Kearns, S. A. Solla, Eds. Cambridge, MA, USA: MIT Press, 1998, pp. 936–942.
[31] M. Corazza and A. Sangalli, "Q-learning and SARSA: A comparison between two intelligent stochastic control approaches for financial trading," Dept. Econ. Res. Paper Series, Univ. Ca'Foscari, Venice, Italy, 2015, vol. 15.
[32] Y. Chen, S. Mabu, K. Hirasawa, and J. Hu, "Genetic network programming with sarsa learning and its application to creating stock trading rules," in *Proc. IEEE Congr. Evol. Comput.*, Sep. 2007, pp. 220–227, doi: 10.1109/CEC.2007.4424475.
[33] J. Moody and M. Saffell, "Reinforcement learning for trading," in *Proc. 11th Int. Conf. Neural Inf. Process. Syst.*, Cambridge, MA, USA: MIT Press, 1998, pp. 917–923.
[34] J. Moody and M. Saffell, "Learning to trade via direct reinforcement," *IEEE Trans. Neural Netw.*, vol. 12, no. 4, pp. 875–889, Jul. 2001, doi: 10.1109/72.935097.
[35] C. Gold, "FX trading via recurrent reinforcement learning," in *Proc. IEEE Int. Conf. Comput. Intell. Financial Eng.*, Mar. 2003, pp. 363–370, doi: 10.1109/CIFER.2003.1196283.
[36] J. Zhang and D. Maringer, "Indicator selection for daily equity trading with recurrent reinforcement learning," in *Proc. 15th Annu. Conf. Companion Genetic Evol. Comput.*, Jul. 2013, pp. 1757–1758, doi: 10.1145/2464576.2480773.
[37] J. Zhang and D. Maringer, "Using a genetic algorithm to improve recurrent reinforcement learning for equity trading," *Comput. Econ.*, vol. 47, no. 4, pp. 551–567, Apr. 2016, doi: 10.1007/s10614-015-9490-y.
[38] W. Si, J. Li, P. Ding, and R. Rao, "A multi-objective deep reinforcement learning approach for stock index future's intraday trading," in *Proc. 10th Int. Symp. Comput. Intell. Design (ISCID)*, Dec. 2017, pp. 431–436, doi: 10.1109/ISCID.2017.210.
[39] Q. Huang, Z. Kong, Y. Li, J. Yang, and X. Li, "Discovery of trading points based on Bayesian modeling of trading rules," *World Wide Web*, vol. 21, no. 6, pp. 1473–1490, Nov. 2018, doi: 10.1007/s11280-018-0534-9.
[40] L. Chen and Q. Gao, "Application of deep reinforcement learning on automated stock trading," in *Proc. IEEE 10th Int. Conf. Softw. Eng. Service Sci. (ICSESS)*, Oct. 2019, pp. 29–33, doi: 10.1109/ICSESS47205.2019.9040728.
[41] S. Chakraborty, "Capturing financial markets to apply deep reinforcement learning," 2019, *arXiv:1907.04373*.
[42] M. Corazza, G. Fasano, R. Gusso, and R. Pesenti, "A comparison among Reinforcement Learning algorithms in financial trading systems," Dept. Econ. Res. Paper Ser., Univ. Ca'Foscari, Venice, Italy, 2019, vol. 33.
[43] Y. Li, P. Ni, and V. Chang, "Application of deep reinforcement learning in stock trading strategies and stock forecasting," *Computing*, vol. 102, no. 6, pp. 1305–1322, Jun. 2020, doi: 10.1007/s00607-019-00773-w.
[44] X. Wu, H. Chen, J. Wang, L. Troiano, V. Loia, and H. Fujita, "Adaptive stock trading strategies with deep reinforcement learning methods," *Inf. Sci.*, vol. 538, pp. 142–158, Oct. 2020, doi: 10.1016/j.ins.2020.05.066.
[45] Y. Shi, W. Li, L. Zhu, K. Guo, and E. Cambria, "Stock trading rule discovery with double deep Q-network," *Appl. Soft Comput.*, vol. 107, Aug. 2021, Art. no. 107320, doi: 10.1016/j.asoc.2021.107320.
[46] M. Taghian, A. Asadi, and R. Safabakhsh, "Learning financial asset-specific trading rules via deep reinforcement learning," *Expert Syst. Appl.*, vol. 195, Jun. 2022, Art. no. 116523, doi: 10.1016/j.eswa.2022.116523.
[47] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep direct reinforcement learning for financial signal representation and trading," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 3, pp. 653–664, Mar. 2017, doi: 10.1109/TNNLS.2016.2522401.
[48] F. Liu, Y. Li, B. Li, J. Li, and H. Xie, "Bitcoin transaction strategy construction based on deep reinforcement learning," *Appl. Soft Comput.*, vol. 113, Dec. 2021, Art. no. 107952, doi: 10.1016/j.asoc.2021.107952.
[49] Z. Zhang, S. Zohren, and S. Roberts, "Deep reinforcement learning for trading," *J. Financial Data Sci.*, vol. 2, no. 2, pp. 25–40, Apr. 2020, doi: 10.3905/jfds.2020.1.030.
[50] Y. Li, W. Zheng, and Z. Zheng, "Deep robust reinforcement learning for practical algorithmic trading," *IEEE Access*, vol. 7, pp. 108014–108022, 2019, doi: 10.1109/ACCESS.2019.2932789.
[51] Y. Li, P. Liu, and Z. Wang, "Stock trading strategies based on deep reinforcement learning," *Sci. Program.*, vol. 2022, pp. 1–15, Mar. 2022, doi: 10.1155/2022/4698656.
[52] F. Khemlichi, H. Chougrad, Y. I. Khamlichi, A. El Boushaki, and S. El Haj Ben Ali, "A stock trading strategy based on deep reinforcement learning," in *Advanced Intelligent Systems for Sustainable Development (AI2SD)*, J. Kacprzyk, V. E. Balas, and M. Ezziyyani, Eds. Cham, Switzerland: Springer, 2022, pp. 920–928, doi: 10.1007/978-3-030-90639-9_74.
[53] R. S. Sutton and A. G. Barto, *Reinforcement Learning, An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

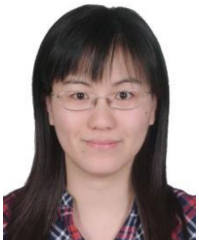
KAI CUI was born in Shanxi, China, in 1995. He received the B.S. and M.S. degrees in electronic science and technology from the North University of China, in 2021. He is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, Macau University of Science and Technology, Macau, SAR, China. His research interests include deep learning, object detection, and computer vision.

RUIZHE HAO was born in Shanxi, China, in 1992. He received the B.S. and M.S. degrees in control science and engineering from the National University of Defense Technology, in 2017. His research interests include knowledge graphs, data governance, blockchain, and privacy-preserving computing.

YULING HUANG received the B.S. degree in information and computing science from Beijing Normal University, Zhuhai, China, and the M.S. degree in mathematics from the Faculty of Science and Technology, University of Macau, Macau, SAR, China. She is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, Macau University of Science and Technology, Macau. Her research interests include deep reinforcement learning, algorithm trading, and machine learning in quantitative investment.

JIANQING LI (Senior Member, IEEE) was born in Shanxi, China. He received the M.S. and Ph.D. degrees in electronic engineering from the Beijing University of Posts and Telecommunications, in 1993 and 1998, respectively.

Since 2004, he has been a Professor with the Macau University of Science and Technology.



YUNLIN SONG received the B.S. degree in mathematics from Nanjing Normal University and the M.S. degree in business administration from the Macau University of Science and Technology, in 2006.

Since 2007, she has been a Lecturer with the Macau University of Science and Technology.