Neurocomputing 449 (2021) 290–302

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

A parallel multi-module deep reinforcement learning algorithm for stock trading

Cong Ma, Jiangshe Zhang *, Junmin Liu, Lizhen Ji, Fei Gao

School of Mathematics and Statistics, Xi'an Jiaotong University, No. 28, Xianning West Road, Xi'an, Shannxi, PR China

ARTICLE INFO

Article history:
Received 21 July 2020
Revised 10 January 2021
Accepted 2 April 2021
Available online 06 April 2021

Keywords:
Parallel multi-module
Reinforcement learning
Capital asset pricing model
Long short-term memory

ABSTRACT

In recent years, deep reinforcement learning (DRL) algorithm has been widely used in algorithmic trading. Many fully automated trading systems or strategies have been built using DRL agents, which integrate price prediction and trading signal generation in one system. However, the previous agents extract the current state from the market data without considering the long-term market historical trend when making decisions. Besides, plenty of related and useful information has not been considered. To address these two problems, we propose a novel model named *Parallel Multi-Module Deep Reinforcement Learning* (PMMRL) algorithm. Here, two parallel modules are used to extract and encode the feature: one module employing Fully Connected (FC) layers is used to learn the current state from the market data of the traded stock and the fundamental data of the issuing company; another module using Long Short-Term Memory (LSTM) layers aims to detect the long-term historical trend of the market. The proposed model can extract features from the whole environment by the above two modules simultaneously, taking the advantages of both LSTM and FC layers. Extensive experiments on China stock market illustrate that the proposed PMMRL algorithm achieves a higher profit and a lower drawdown than several state-of-the-art algorithms.

1. Introduction

Algorithmic trading generates trading signals and executes orders using a computer program, which follows a predefined trading strategy or trading logic. The automated trading system has many superiorities to human traders, such as, more stable system, faster order execution, and not being affected by the emotional factor. Thus, algorithmic trading has become a valuable topic in the modern finance field in recent years.

With the success of artificial intelligence in many fields, using machine learning algorithms in financial trading [1–3] has attracted widespread attention, such as stock selection, price prediction, portfolio management, and risk management. Many machine learning algorithms have been used to improve the traditional trading strategies, including random forest [4,5], support vector machine (SVM) [6,7] and neural network (NN) [8,9] algorithms. But these supervised learning algorithms can only select stocks or predict price but cannot build an entire automated trading strategy.

As the Alpha Go beats the human professional player in 2015 [10], deep reinforcement learning (DRL) algorithm has received widespread attention from academia and industry [11–15], which combines the advantages of deep learning and reinforcement learning (RL). The stock trading problem can be regarded as a Markov Decision Process (MDP), and the optimal strategy can be solved by dynamic programming under the DRL framework. Thus, some researchers have built some automated trading strategies using the DRL algorithm. For example, Neuneier et al. [16] firstly use RL algorithm in Foreign Exchange, getting better performance than the previous supervised learning algorithms. Moody et al. [17] propose a recurrent reinforcement learning (RRL) algorithm to train the trading system for portfolios by optimizing the objective function, where the recurrent neural network (RNN) is used to encode state from the trading environment. Dempster et al. [18] build an automated trading system based on an adaptive RL algorithm, getting a consistent return while avoiding a large drawdown. Deng et al. [19] propose a task-aware backpropagation through time method to efficiently train the agent and represent the real-time financial signal by RNN, which can extract higher-level features from the state and get reliable profit on China future market. Rundo et al. [20] adopt RL algorithm with long short-term memory

* Corresponding author.
E-mail address: jszhang@mail.xjtu.edu.cn (J. Zhang).

https://doi.org/10.1016/j.neucom.2021.04.005
0925-2312/

(LSTM) layer to predict the medium-short trend and maximize the return in Foreign Exchange Market.

Although the above algorithms have a good performance in several security markets, the agent learns features from the market data and doesn't make full use of other available information or knowledge, such as fundamental data or some financial engineering models. Specifically, they do not consider the theoretical financial model, especially the positive relationship between risk and return [21]. The classical capital asset pricing model (CAPM) [22] points out that the expected return on a security is equal to the risk-free return plus a risk premium, which can be expressed as

$$R - R_f = \beta(R_m - R_f) + \epsilon, \tag{1}$$

where R is the expected return of the portfolio, R_f is the risk-free return, and R_m is the expected return of the market. β is the systematic risk coefficient, that is, the fluctuation of its price change relative to the overall market. It can be calculated by

$$\beta = \frac{Cov(R, R_m)}{Var(R_m)}.$$

According to the CAPM, the return is linearly related to the systematic risk of the market. Based on the previous work [23], systematic risk is taken by the impact of economic, geopolitical, and financial factors, which is a powerful determinant of return. These impacts will be mainly reflected in the price of related stocks. Thus, the agent should consider these price sequences when taking action.

Furthermore, from the later studies [24–26], ϵ can be regarded as unsystematic risk, usually denoted as α, which is inherent in a specific company or industry and embodied in the fundamental data. It can be calculated as the following

$$\alpha = R - R_f - \beta(R_m - R_f).$$

Hence, the issuing company's fundamental data, including capitalization, book-to-market value, and so on, has an essential influence on the expected return. As α and β can only be calculated by the return after trading, we can introduce some useful information to build the trading strategy. Thus, the agent needs to consider the related stocks' price series and the issuing company's fundamental data when taking action.

In this paper, considering the above factors, we propose a parallel multi-module deep reinforcement learning (PMMRL) algorithm for stock trading. In this algorithm, one module exploiting the fully connected (FC) layer learns the current state from market data and fundamental data of the traded stock. Another module using the LSTM layer learns the market's long-term historical trend feature from the past price series. These price series contain the traded stock price sequence and the related stocks' price series to represent the whole market trend. In this way, the agent considers both the current state and historical trend when taking action. Extensive experiments implemented in the China Stock Market show the proposed algorithm yields higher profit and undertakes lower risk than several existing algorithms. In summary, the main contributions of this paper are summarized as follows:

(1) A novel DRL algorithm named PMMRL is proposed for stock trading. The agent adopts two parallel modules to extract and encode the feature from the environment: LSTM module aims to detect the long-term historical trend of the market, and FC module is used to encode the current state. In this parallel way, the agent can efficiently encode the related information of the financial market environment.
(2) Experimental results on the China Stock Market show that the performance of the PMMRL algorithm has exceeded several state-of-the-art algorithms based on several evaluation criteria. Significantly, the cumulative return ratio has increased by more than 200% in the best case. Further, the ablation studies in Section 4.3.3 have illustrated the effectiveness of the 'parallel multi-module' mechanism.

The rest of this paper is organized as follows. Section 2 introduces several preliminary concepts and algorithms. Section 3 illustrates the proposed PMMRL algorithm. In Section 4, we implement many numerical experiments and compare the proposed model with several state-of-the-art algorithms. Section 5 concludes this paper.

2. Preliminaries

In this section, we introduce some preliminaries about MDP and double deep Q-learning network (DDQN) algorithm.

2.1. Markov decision process

Stock trading can be regarded as an MDP where an agent interacts with the financial market by buying or selling stocks and gets the corresponding reward from the market, denoted as $M = (\mathscr{S}, \mathscr{A}, \mathscr{T}, \gamma, R)$. Here, $\mathscr{S}$ is the state space, $\mathscr{A}$ is the action space, $\mathscr{T}$ is the transition probabilities, $\gamma \in (0, 1]$ is the discount factor, and R is the immediate reward. That is, given a state $s_t \in \mathscr{S}$, the agent will take an action $a_t \in \mathscr{A}$ and arrive at a new state $s_{t+1} \in \mathscr{S}$ according to the probability $P_{s_t,a_t}(s_{t+1}) \in \mathscr{T}$. Meanwhile, the agent will get the correponding reward $R(s_t, a_t, s_{t+1})$. These interactions between the agent and financial environment will produce a trajectory $\tau = \{s_0, a_0, R_0, s_1, a_1, R_1, s_2, a_2, R_2, \ldots\}$. The objective function for this process is to maximize the expected cumulative return at each time step T, which is given as

$$J_T = \sum_{t=T}^{+\infty} \gamma^{t-T} R_t, \tag{2}$$

where R_t is the immediate reward at time t.

2.2. Double deep Q-network algorithm

In this paper, we apply the *double deep Q-network* (DDQN) [27] algorithm to train the agent and get the optimal policy. *Q-learning* algorithm [28] is to learn a greedy deterministic policy that takes the optimal action given the current state. Specifically, Q-learning seeks to learn a policy that maximizes the total reward. It defines a function to measure the state-action combination

$$Q : \mathscr{S} \times \mathscr{A} \to \mathbb{R},$$

which can be formulated as

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma \max_a Q(s_{t+1}, a), \tag{3}$$

where $r(s_t, a_t)$ is the reward of taking action a_t at the state s_t, $Q(s_t, a_t)$ is an action-value function, that is, the return when taking action a_t at the state s_t. The equation is always called the Bellman equation, which is the core of Q-learning algorithm. Here, the replay memory $D = \{(s_t, a_t, r_t, s_{t+1})\}_t$ is created to train the agent, which stores Q-values for all possible combinations of state s and action a. But as the limitation of computation resources, we only store the last N experiences in practice.

Deep Q-network (DQN) algorithm [11] is an improved version of *Q-learning* algorithm, where Q-function $Q_\theta(s, a)$ is estimated by a deep network, θ is the parameter of the model. The objective function of DQN algorithm is

$$\min_\theta J(\theta) = \min_\theta (y_t - Q_\theta(s_t, a_t))^2, \tag{4}$$

where y_t is calculated as follows

$$y_t = \begin{cases} r_t, & \text{if episode terminates at step } t+1 \\ r_t + \gamma \max_a Q_{\theta^-}(s_{t+1}, a'), & \text{otherwise} \end{cases}, \tag{5}$$

where (s_t, a_t, r_t, s_{t+1}) is sampled from the replay memory D randomly. The target network with parameter θ is the same as the online network, and the parameter of the model will be copied every τ steps from the online network, that is, $\theta_t = \theta_t^-$. Besides, the second equation of Eq. (5) can be written as

$$y_t^{DQN} = r_{t+1} + \gamma Q_\theta(s_{t+1}, \text{argmax}_a Q_\theta(s_{t+1}, a)). \tag{6}$$

From this equation, we can see that DQN algorithm adopts the same values both to select and evaluate an action, which is more likely to result in over-optimistic value estimation. Furthermore, in order to overcome this problem and enhance the stability of the Q values, DDQN algorithm [27] is proposed to decouple the action selection and value estimation by adopting two deep networks Q_θ and Q'_θ, where Q_θ is used to select the best action at state s' and Q'_θ is used to evaluate the value of this policy, and θ and θ' are the parameters of the two networks, as shown in Fig. 2(a). In this way, Eq. (6) can be written as

$$y_t^{DDQN} = r_{t+1} + \gamma Q'_\theta(s_{t+1}, \text{argmax}_a Q_\theta(s_{t+1}, a)). \tag{7}$$

This change can improve the stability and performance when learning complicated task.

3. Parallel multi-module deep reinforcement learning algorithm

In this section, we will illustrate the framework of the proposed PMMRL algorithm for stock trading. Firstly, we introduce three critical components of this model: state space, action space, and reward function. Then, we illustrate the proposed model in detail.

3.1. State space

The state we used in this paper includes the following five parts:

1) Current market feature $S_{1,t} = [High_t, Open_t, Low_t, Close_t, Volume_t, MA_t, MACD_t, RSI_t, OBV_t]$, where $High_t, Open_t, Low_t, Close_t, Volume_t$ are high, open, low, close prices and volume of traded stock at time t, respectively. $MA_t, MACD_t, RSI_t, OBV_t$ (see Appendix A) are the technical indicators at the time t.

2) Current fundamental data $S_{2,t}$ of the traded stock, which contains 14 fields, including total profit, market capitalization, circulating market capitalization, turnover ratio, total operating revenue, total operating cost, total profit, net profit, price-to-earnings ratio, price-to-book ratio, basic earnings per share, return on equity, return on assets, total composite income. This feature reflects the company's value.

3) Long-term historical trend $S_{3,t} = \{P_{t,i}^k\}_{i=0}^n$, which is the historical close price sequences of the traded stock and the related stocks, where n is the number of total stocks, $P_{t,i}^k$ is the close price sequence of stock i with the lookback period k at time t. Considering the computation speed, the lookback period k is set as 30 trading days according to the grid search.

4) The amount of stock holding $h_t \in 100 * \mathbb{Z}$, where $\mathbb{Z}$ is the integer numbers and 100 is the trading unit of China stock market. The introduction of h_t can make the agent aware of the previous transactions in order to avoid frequent transactions and reduce the transaction fees.

5) The unrealized profit and loss (uPNL) $b_t \in \mathbb{R}_+$, where $\mathbb{R}_+$ is non-negative real numbers. It is introduced because an investor would look at the uPNL before closing positions, which can be calculated as follows

$$b_t = \begin{cases} (p_t - p_{buy})/p_{buy}, & h_t > 0 \\ 0, & else \end{cases},$$

where p_t is the close price at the current time t, and p_{buy} is the buying price of the current position.

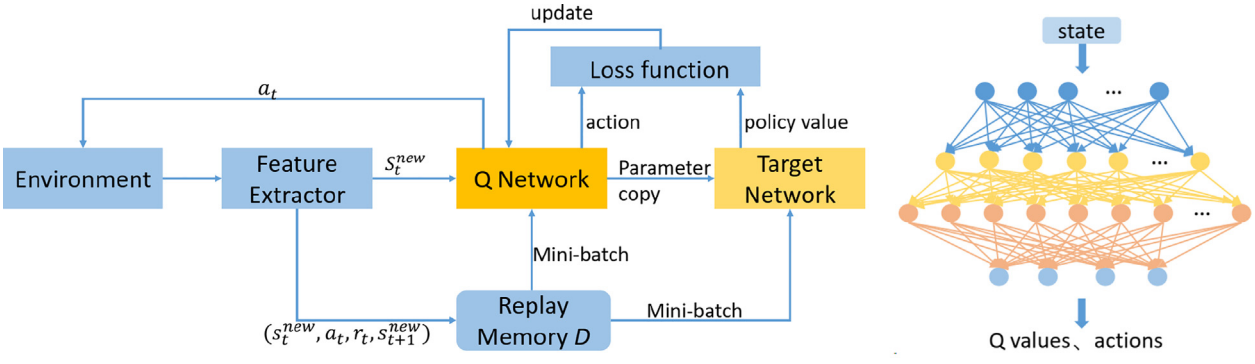
3.2. Action space

The available actions contain buying, selling, and holding, which can be denoted as $a \in \{+1, -1, 0\}$, respectively. When the action is + 1 and the position is 0, the agent will spend all the cash to buy stock. When the action is −1 and the position is greater than 0, the agent will sell all the positions. When the action is 0, the agent will not take any action. There is no 'add position' or 'short position' actions.

3.3. Reward

The reward of taking an action for the agent at time t is

$$r_t = \begin{cases} -f_1, & a_t = 1 \\ p_t - p_{buy} - f_2, & a_t = -1 \\ p_t - p_{buy}, & a_t = 0, h_t > 0 \end{cases},$$



(a) DDQN algorithm (b) Q-network and target network

Fig. 2. The architecture of *DDQN* algorithm.

p_t is the close price at the current time t, p_{buy} is the buying price of the current position, and f_1, f_2 is the transaction fees of buying, selling stocks, respectively.

3.4. The architecture of PMMRL algorithm

Inspired by the theoretical finance models [22,26,24,25], especially the CAPM, the expected return of one portfolio is not only related to its price series but also related to the fundamental data of the issuing company and the price sequences of the related stocks. Thus, it is vital for the agent to introduce and encode useful information reasonably.

Considering the characteristics of different data, we introduce a parallel structure into the RL algorithm and propose the Parallel Multi-Module Deep Reinforcement Learning (PMMRL) algorithm, where FC and LSTM modules are used to extract and encode features from the environment simultaneously. That is, FC module extracts the current state of the traded stock, represented as $\mathcal{S}_{current} = FC(S_1, S_2)$. Besides, stocks from one industry are affected by similar economic, geopolitical, and financial factors, thus we extract the relevant feature with systematic risk from these price sequences of stocks. Here, LSTM module is used to extract long-term historical trend of the overall industry, denoted as $\mathcal{S}_{history} = LSTM(S_3)$. Then, the agent get the new state $\mathcal{S}^{new} = \{\mathcal{S}_{current}, \mathcal{S}_{history}, h_t, b_t\}$ by concatenating $\mathcal{S}_{current}, \mathcal{S}_{history}, h_t$ and b_t, as shown in Fig. 1. h_t and b_t are concatenated to make the agent realize the current position and uPNL. In this way, the agent can not only find out the current market state but also the long-term historical trend when taking action. With the whole state of current environment, the agent can find the optimal policy by continuously interacting with the environment. The pseudocode of the proposed *PMMRL* algorithm is presented in Algorithm 1.

Algorithm 1. *Parallel Multi-Module Deep Reinforcement Learning (PMMRL) Algorithm for Stock Trading*

Input: Environment *Env*
Output: policy
Initialize replay memory D with capacity N
Initialize behaviour network Q_θ
Initialize target network Q'_θ
for each episode i in {1,. . .,M} **do**
  Initialize market feature S_1, fundamental feature S_2, and historical trend feature S_3
  Preprocess states $\mathcal{S}_{current} = FC(S_1, S_2)$ and $\mathcal{S}_{history} = LSTM(S_3)$
  Concatenate $\mathcal{S}_{current}$ and $\mathcal{S}_{history}$ and get the new state $\mathcal{S}^{new} = \{\mathcal{S}_{current}, \mathcal{S}_{history}, h_t, b_t\}$
  for t = 1,. . .,T **do**
    With probability ϵ select a random action a_t, otherwise select $a_t = \text{argmax}_a Q_\theta(\mathcal{S}_t^{new}, a)$
    Execute action a_t in *Env*, get the next state $\mathcal{S}_{t+1}^{new}$ and the corresponding reward r_t
    Save transition $(\mathcal{S}_t^{new}, a_t, r_t, \mathcal{S}_{t+1}^{new})$ into replay memory D
    Sample minibatch of transitions $\{(\mathcal{S}_i^{new}, a_i, r_i, \mathcal{S}_{i+1}^{new})\}_{i=1}^{N}$ from replay memory D randomly
    Calculate

$$y_i = \begin{cases} r_i, & \text{if episode terminates at step } i+1 \\ r_i + \gamma \max_a Q'_\theta(s_{i+1}, \text{argmax}_a Q_\theta(s_{i+1}, a)), & \text{otherwise} \end{cases},$$

    Perform a gradient descent on $(y_i - Q_\theta(\mathcal{S}_i^{new}, a_i))^2$ with respect to the network parameter θ
    Every C steps, $\theta' \leftarrow \theta$
  end for
end for

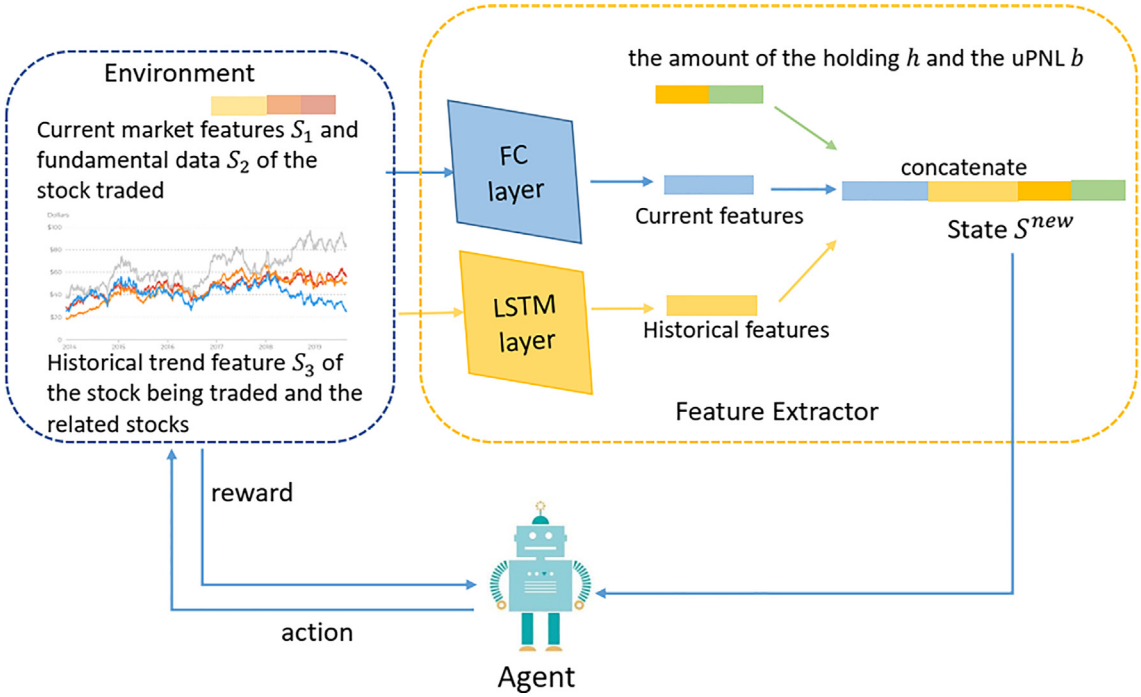


Fig. 1. The architecture of *PMMRL* algorithm.

4. Experiment

In this part, we compare the proposed PMMRL algorithm with several state-of-the-art algorithms by implementing many numerical experiments on China Stock Market based on Tensorflow [29] to illustrate its effectiveness and superiority.

4.1. Dataset and experimental setting

We collect the daily market data of stocks and the fundamental data of the issuing company from the JointQuant [30], which is an online quantitative trading platform, providing all stocks data of Shanghai (SH) and Shenzhen (SZ) Stock Exchanges from 2005 to present. We randomly select several stocks from different industries to trade, as shown in Table 1. The data we used contains two parts: one is the current state, including the market data and fundamental data of the traded stock; another is historical price sequences of the related stocks, reflecting the long-term market trend. These related stocks are selected from one industry according to *Pearson correlation coefficient* of daily return shown in the Table 1. From Fig. 3, we can see that their trends are somewhat similar and also slightly different. Some stocks are more sensitive to market fluctuation and react earlier. Thus, it's very reasonable that using historical price sequences of the related stocks represents the historical market trend. Market data contains high, open, low, close prices, volume, and several technical indicators (see Appendix A). The fundamental data from the financial statements of the company contains 14 fields such as price-to-earnings ratio, price-to-book ratio, turnover ratio, market capitalization, circulating market capitalization, total operating revenue, and so on (introduced in Section 3.1). Here, the units of the above variables are Chinese yuan except for ratio and volume. The unit of volume is share.

The whole data is divided into two parts: the earlier 75% as the training set and the remaining 25% as the test set. And each agent is trained for 20 episodes in the training set. The initial cash is set as 100,000 yuan. The transaction fees of buying stocks and selling stocks are set as 0.1%, 0.2% of the transaction amount, respectively.

Considering three types of price trends 'up, down, flat', the LSTM module is set as one layer with three hidden neurons. The activation function is set as tanh function, and the activation function for the recurrent step is hard sigmoid. The FC module we used only has one hidden layer with 24 neurons, and its activation function is ReLU. The agent can recognize the current state $\mathscr{S}^{new}$ by the two parallel modules, as shown in Fig. 1. DDQN algorithm is used to train the agent, as depicted in Fig. 2(a). Here, the Q-network and target network of DDQN algorithm are set as FC network, as shown in Fig. 2(b). There are four layers with 24, 48, 96, 4 neurons to learn the reward and action functions from the state $\mathscr{S}^{new}$. The activation function for the first three layers is set as ReLU, and the activation function for the last layer is linear. The capacity N of replay memory D is set as 3000. The agent is trained by Adam optimizer with a learning rate of 0.001 based on the Tensorflow [29].

4.2. Baseline methods

.

- **Buy and Hold (B&H)** [31] is a passive investment strategy, where an investor buys the stock at the start time and holds them until the end of the investment, regardless of short-term fluctuations.
- **Double Exponential Moving Average (DEMA)** [32] is a classical trading strategy that is implemented based on the relationship between DEMA value and price. This indicator can be formulated as

$$DEMA_t = 2 \times EMA_t^k(p_t) - EMA_t^k(EMA_t^k(p_t)),$$

where k is the look-back period and set as 14, $EMA_t^k(p_t)$ is the exponential moving average value of close price p_t with k look-back days. When the price is above (below) DEMA value, the uptrend (downtrend) is confirmed, then the investor will buy (sell) the stock.
- **Fuzzy deep direct reinforcement learning (FDDR)** algorithm [19] extracts the features using the combination of fuzzy learning and neural network, as shown in Fig. 4(a). The input features have 50 dimensions, including the close price of the last 45 days and the momentum change to the previous 3, 5, 7, 15, and 30 days. The membership function of the fuzzy representation part (purple panel) is initialized by the means and variances of centroid calculated using k-means. Through this part, 50 input nodes are converted into 150 nodes of first-level fuzzy representation. There are four FC layers in the deep transformation part, with 128, 128, 128, and 20 hidden nodes per layer. The corresponding activation functions are set as sigmoid. The agent is trained by task-aware BPTT proposed by the authors.
- **Reinforcement learning with price trailing (RLPT)** algorithm [33] proposes a novel state input and a reward function for the agent. Each state from the environment is defined as

$$S_t = [p_t - h_t + m, h_t - p_t + m, a_{t-1}] \in \mathbb{R}^5, \tag{8}$$

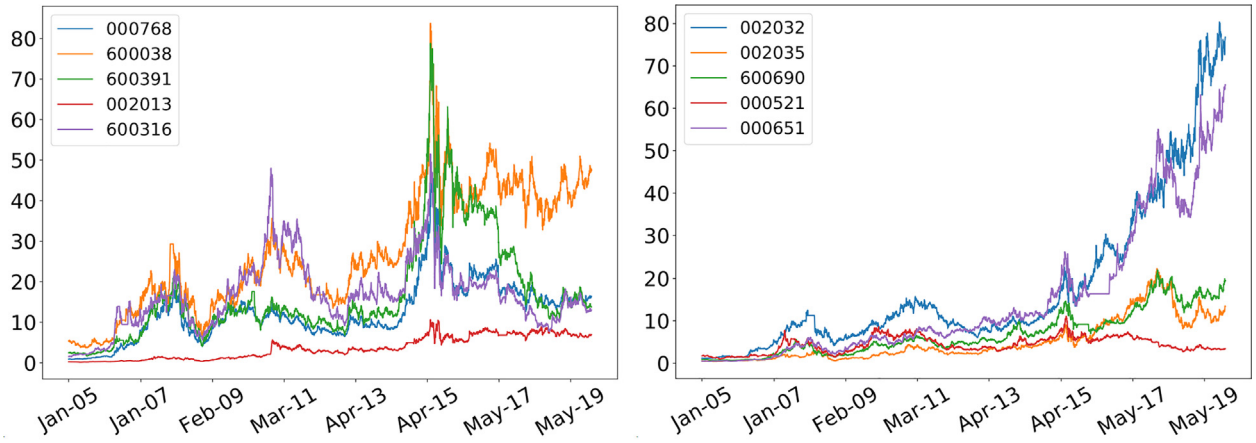
where p_t is close price at time t, h_t is the position of time t, m is a margin value and set as 0.01, $a_{t-1} \in \{[1,0,0],[0,1,0],[0,0,1]\}$ is the action performed of the last step, the first two values are the distance between the current estimation and the upper and lower margin of the agent. The position-based reward is defined as

$$r_t^{(p)} = \begin{cases} \frac{h_t - p_t + m}{m}, & if \ \ h_t \leqslant p_t \\ \frac{p_t - h_t + m}{m}, & otherwise \end{cases}.$$

The final reward is calculated by

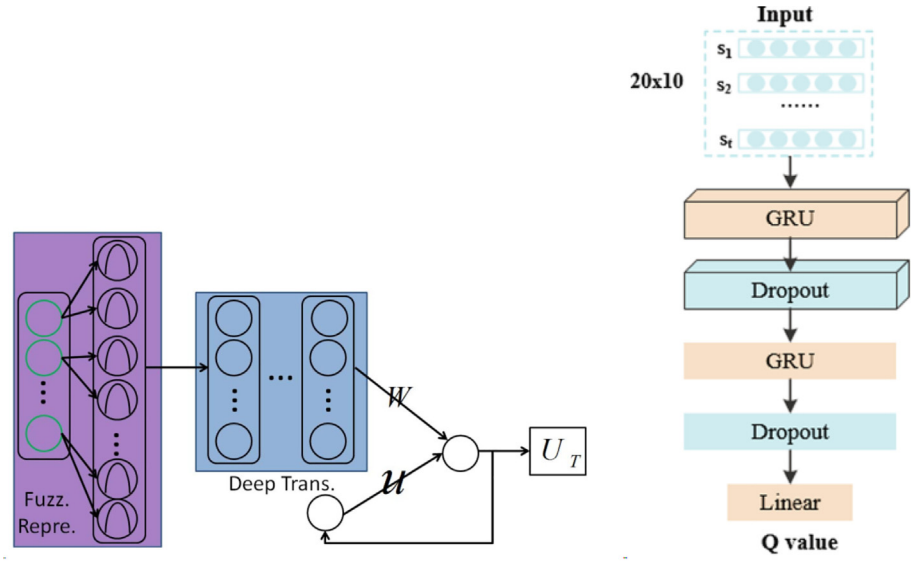
Table 1
Stock information.

Stock traded	Sector/Stock Name	Relative stocks	Training period	Test period
000568.SZ	Chinese liquor/ Luzhou Laojiao	000858.SZ, 600519.SH, 600809.SH, 000860.SZ	2005/01/01–2016/04/19	2016/04/20–2019/12/31
000768.SZ	Aviation/ AVIC Aircraft	600038.SH, 600391.SH, 002013.SZ, 600316.SH	2007/01/01–2016/10/10	2016/10/11–2019/12/31
000860.SZ	Chinese liquor/ Beijing Shunxin Agriculture	000858.SZ, 600809.SH, 000568.SZ, 000799.SZ	2005/01/01–2016/04/19	2016/04/20–2019/12/31
002032.SZ	Appliance/ Zhejiang Supor	002035.SZ, 600690.SH, 000521.SZ, 000651.SZ	2005/01/01–2016/04/19	2016/04/20–2019/12/31
600000.SH	Bank/ Shanghai Pudong Development Bank	600036.SH, 600015.SH, 600016.SH, 000001.SZ	2005/01/01–2016/04/19	2016/04/20–2019/12/31
600036.SH	Bank/ China Merchants Bank	600000.SH, 600016.SH, 600015.SH, 000001.SZ	2005/01/01–2016/04/19	2016/04/20–2019/12/31
600498.SH	Technology/ Fiberhome Telecom Tech	600522.SH, 600487.SH, 000988.SZ, 600776.SH	2005/01/01–2016/04/19	2016/04/20–2019/12/31
601628.SH	Insurance/ China Life Insurance Company	601601.SH, 601318.SH, 600291.SH, 000627.SZ	2008/01/01–2016/12/30	2017/01/03–2019/12/31



(a) 000768.SZ and its related stocks (b) 002032.SZ and its related stocks

Fig. 3. The daily close price of the traded stock and related stocks. The horizontal axis is the date and the vertical axis is the price.



(a) FDDR algorithm [19] (b) GDQN algorithm [34]

Fig. 4. The frameworks of several baseline algorithms.

$$r_t = \begin{cases} r_t^{(p)} - |r_t^{(p)}| \cdot f, & if \ \ \delta_t = 1 \\ r_t^{(p)}, & otherwise \end{cases},$$

where f is the commission, $\delta_t = 1$ when the agent changes its price trend estimation from upwards to downwards or vice versa, otherwise $\delta_t = 0$. The agent is trained using the DDQN algorithm. The employed Q-network and target network have the same setting. There are four hidden layers with 64, 64, 64, and 3 nodes, which is similar to the structure of Fig. 2(b). PReLU is used as the activation function of the first three layers, and no function is for the last layer. The network's output is Q-values.

- **Gated Deep Q-learning trading strategy (GDQN)** algorithm [34] combines gated recurrent unit (GRU) with RL algorithm, where DDQN is used to train the agent. The input features include open, close, high, low, volume, and several technical indicators, which are normalized to a 20-dimensional vector. The time step of the feature is 10. There are two same networks in this algorithm: Q network and target network. The network has five layers, as shown in Fig. 4(b). The output feature's size is (10, 32) through the first GRU layer and dropout. The output shape changes to 32 after the second GRU and dropout layers. The output of the last linear layer with three nodes is Q values. The action is decided according to the Q values. The ratio of dropout is 20 percent during the training.

4.3. Experiment results

In this section, we implement many experiments on the China Stock Market to illustrate the proposed PMMRL algorithm's effectiveness. Here, we adopt DDQN algorithm to train the agent. The results of PMMRL and several baselines algorithms are shown in Table 2. Also, Fig. 5 shows the change process of the total assets with different stocks. Besides, due to space limitation, more experiment results are shown in Table C.6 (in the Appendix part).

4.3.1. Performance analysis

According to these results, we can see that the cumulative return ratio of PMMRL algorithm has exceeded other baseline algo-

rithms in almost all stocks. Comparing these results of 000568.SZ in Table 2, the cumulative return of PMMRL algorithm has been raised to 475.89%, which is far beyond the return of other baseline algorithms. Sharpe ratio has been improved a lot, which means that the investor can earn more return per unit of risk. Also, Sortino and Calmar ratios have increased a lot when using the PMMRL algorithm, while MDD and longest DD days decrease in the results of 000568.SZ. Among the results of 000768.SZ, we find that MDD value of the PMMRL algorithm is not the lowest, but the cumulative return and Sharpe ratio have increased a lot by comparing it with several other baseline algorithms. Similar results can be seen from the results of 000860.SZ, 002032.SZ, 600000.SH, and 600036.SH, which fully illustrates the effectiveness of PMMRL algorithm. This table further explains that the proposed PMMRL algorithm can effectively improve the agent's performance and get a more steady profit.

Furthermore, several subfigures depict the total asset's change of several different models in Fig. 5. The initial cash is 100,000 yuan. The B&H curve essentially reflects the change in stock price. And the red dash lines represent the proposed algorithm. It is evident that these red lines are higher than the other curves in almost all figures. No matter how the price fluctuates, these red curves keep rising and rarely drop sharply in all subfigures. Especially, the price of 600000.SH has fallen from the beginning of 2018 in Fig. 5(e). The return curve of the PMMRL algorithm only fluctuates a little, but the return of other baseline algorithms have dropped sharply. It means that the PMMRL algorithm has a more robust performance and performs better than other baseline algorithms in a downward market. Besides, according to these results, we can see that the PMMRL algorithm can get more profit than other baselines when the price goes up. These results further explain PMMRL algorithm is more effective and can gain stable profit no matter the price goes up or down.

4.3.2. Trading signal analysis

In this section, we randomly choose a set of experiments to show the trading points of different algorithms, as shown in

Table 2
The performance evaluation of the proposed PMMRL algorithm and several baseline algorithms on several Chinese stocks.

Stock codes	Methods	Cumulative return ratio	Sharpe ratio	Sortino ratio	Calmar ratio	MDD	Longest DD days
000568.SZ	B&H	302.96%	1.23	1.95	0.9123	−50.16%	448
	DEMA	59.76%	0.64	1.03	0.4096	−32.97%	402
	FDDR	120.68%	0.81	1.25	0.4588	−52.01%	484
	PTRL	−1.43%	0.10	0.14	-0.0088	−43.88%	778
	GDQN	142.41%	1.29	2.18	1.0562	−25.61%	512
	PMMRL	**475.89%**	**1.85**	**3.11**	**2.4753**	**−24.46%**	**245**
000768.SZ	B&H	−22.88%	−0.09	−0.12	−0.1533	−50.51%	993
	DEMA	10.91%	0.26	0.41	0.1365	−23.93%	480
	FDDR	17.34%	0.32	0.44	0.1301	−39.1%	987
	PTRL	−1.74%	−0.04	−0.05	−0.0465	**−11.72%**	664
	GDQN	−21.56%	−0.08	−0.11	−0.1441	−50.40%	993
	PMMRL	**63.04%**	**0.72**	**1.05**	**0.6466**	−25.34%	**266**
000860.SZ	B&H	263.57%	1.11	1.81	1.1550	−36.16%	648
	DEMA	156.61%	1.08	1.94	0.8609	−33.71%	703
	FDDR	144.68%	0.91	1.47	0.7395	−37.01%	**161**
	PTRL	151.04%	0.91	1.49	0.9791	−28.86%	637
	GDQN	50.07%	0.74	1.07	0.4554	**−25.47%**	411
	PMMRL	**371.78%**	**1.40**	**2.36**	**1.5343**	−33.96%	351
002032.SZ	B&H	219.78%	1.09	1.82	1.6861	−21.90%	260
	DEMA	7.58%	0.2	0.32	0.0777	−25.70%	767
	FDDR	124.42%	0.85	1.41	1.0120	−24.14%	330
	PTRL	0.77%	0.07	0.11	0.0275	**−7.59%**	1016
	GDQN	275.10%	1.55	2.87	2.4011	−17.89%	271
	PMMRL	**533.42%**	**1.86**	**3.38**	**4.0783**	−15.87%	**155**
600000.SH	B&H	7.70%	0.20	0.31	0.0619	−32.69%	896
	DEMA	10.02%	0.26	0.41	0.0761	−34.34%	896
	FDDR	12.40%	0.27	0.41	0.1070	−30.01%	701
	PTRL	−6.48%	-0.60	-0.73	-0.2440	**−7.35%**	1334
	GDQN	21.20%	0.39	0.61	0.1672	−31.91%	542
	PMMRL	**97.89%**	**1.46**	**2.47**	**1.9494**	−10.40%	**196**
600036.SH	B&H	148.89%	1.14	1.79	0.9965	−28.06%	416
	DEMA	94.23%	1.11	1.83	1.2332	−15.94%	**167**
	FDDR	16.17%	0.33	0.47	0.1616	−25.59%	693
	PTRL	111.94%	0.99	1.54	0.8417	−26.75%	416
	GDQN	80.69%	1.14	1.84	1.5836	**−10.95%**	301
	PMMRL	**365.95%**	**2.03**	**3.35**	**1.8424**	−28.01%	381
600498.SH	B&H	21.49%	0.33	0.47	0.1382	−39.15%	777
	DEMA	−26.54%	−0.24	−0.35	−0.2188	−36.60%	1208
	FDDR	35.40%	0.43	0.60	0.3127	−27.30%	525
	PTRL	10.59%	0.26	0.36	0.1183	−23.32%	774
	GDQN	−0.55%	0.11	0.16	−0.0057	−26.32%	628
	PMMRL	**394.39%**	**1.62**	**2.53**	**2.5781**	**−20.98%**	**111**
601628.SH	B&H	44.82%	0.58	0.89	0.3222	−40.89%	712
	DEMA	34.86%	0.58	1.0	0.3011	−34.91%	602
	FDDR	69.32%	**0.97**	**1.51**	**0.9201**	**−20.92%**	497
	PTRL	67.40%	0.86	1.36	0.6786	−27.69%	**469**
	GDQN	37.03%	0.51	0.79	0.2698	−41.16%	713
	PMMRL	**69.66%**	0.78	1.23	0.4671	−41.37%	694

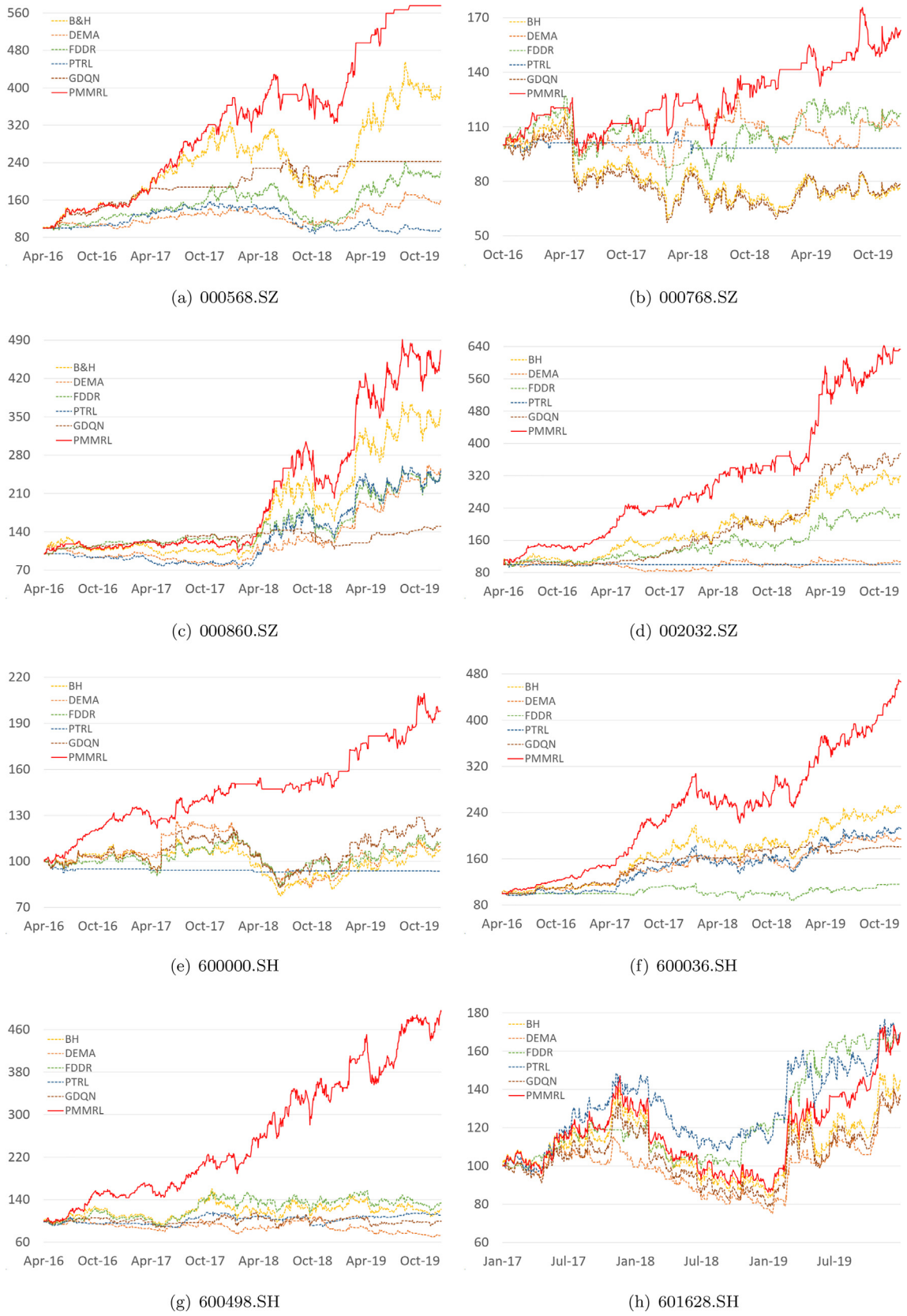


Fig. 5. The total asset change processes of different algorithms. The horizontal axis is the date. The vertical axis is the total asset, and the unit is 1000 yuan.

Fig. 6. By comparing Fig. 6(a) with Fig. 6(d), we find that there are many trading points in the congestion area. Meanwhile, referring to the red box of the FDDR algorithm, these tradings cause a considerable loss. Similar transactions appear in PTRL and GDQN algorithms, shown in Figs. 6(b) and 6(c). The stock price of 000568.SZ drops sharply around July 2018, but the PTRL agent doesn't recog-

Table C.6
The performance evaluation of the proposed PMMRL algorithm and several baseline algorithms on more Chinese stocks.

Stock codes	Methods	Cumulative return	Sharpe ratio	Sortino ratio	Calmar ratio	MDD	Longest DD days
000559.SZ	B&H	−55.79%	−0.59	−0.82	−0.3051	−64.91%	1208
	DEMA	−24.06%	−0.24	−0.37	−0.1717	−41.77%	1301
	FDDR	7.85%	**0.40**	**0.64**	**0.365**3	**−5.65%**	1010
	PTRL	−55.86%	−0.60	−0.82	−0.3056	−64.91%	1208
	GDQN	−54.98%	−0.60	−0.83	−0.3018	−64.31%	1208
	PMMRL	**9.83%**	0.26	0.42	0.1360	−18.89%	**251**
000630.SZ	B&H	−14.53%	−0.04	−0.06	−0.0932	−44.61%	1049
	DEMA	−6.76%	−0.01	−0.01	−0.0477	−39.29%	1142
	FDDR	−14.68%	−0.04	−0.06	−0.0942	−44.61%	1049
	PTRL	7.88%	0.21	0.31	0.0903	**−22.95%**	**882**
	GDQN	−4.06%	0.08	0.12	−0.0250	−44.64%	1049
	PMMRL	**13.15%**	**0.26**	**0.40**	**0.1050**	−32.35%	1049
000785.SZ	B&H	−15.96%	0.08	0.11	−0.0745	−61.67%	1247
	DEMA	12.92%	0.26	0.40	0.0861	−38.78%	930
	FDDR	−9.55%	0.13	0.18	−0.0436	−61.43%	1247
	PTRL	42.78%	0.47	0.70	0.1903	−53.12%	964
	GDQN	−17.37%	0.06	0.09	−0.0858	−58.59%	1128
	PMMRL	**180.04%**	**1.26**	**1.95**	**1.3551**	**−23.69%**	**196**
000799.SZ	B&H	137.28%	0.77	1.23	0.4951	−53.15%	661
	DEMA	142.96%	0.96	1.71	0.9379	**−28.92%**	**307**
	FDDR	−10.48%	-0.04	-0.05	-0.0677	−43.57%	1132
	PTRL	234.42%	1.13	1.98	0.9597	−40.22%	580
	GDQN	97.18%	0.66	1.03	0.3800	−53.03%	661
	PMMRL	**442.83%**	**1.44**	**2.43**	**1.8633**	−31.12%	389
000895.SZ	B&H	18.62%	0.42	0.62	0.3301	−26.65%	329
	DEMA	−20.60%	−0.38	−0.51	−0.2600	−41.42%	574
	FDDR	−30.99%	−0.64	−0.81	−0.3848	−43.50%	649
	PTRL	−21.73%	−0.35	−0.49	−0.3563	−31.99%	574
	GDQN	19.75%	0.57	0.81	0.6021	**−15.46%**	**169**
	PMMRL	**20.95%**	**0.60**	**0.85**	**0.6290**	−15.66%	347
600104.SH	B&H	47.66%	0.54	0.80	0.3268	−34.01%	651
	DEMA	21.09%	0.38	0.56	0.2889	−18.38%	449
	FDDR	21.62%	0.34	0.49	0.1398	−38.86%	651
	PTRL	50.48%	0.63	0.91	0.3386	−34.50%	701
	GDQN	16.03%	0.78	**1.34**	**1.0729**	**−3.82%**	**368**
	PMMRL	**82.15%**	**0.83**	1.24	0.5854	−30.07%	553
600221.SH	B&H	−45.11%	−0.79	−1.03	−0.3257	−52.12%	991
	DEMA	−30.48%	−0.67	−0.94	−0.3174	−33.59%	1138
	FDDR	−29.21%	−0.87	−1.05	−0.2578	−39.40%	986
	PTRL	−41.73%	−0.85	−1.07	−0.3279	−47.07%	1050
	GDQN	−48.50%	−1.22	−1.45	−0.3414	−54.47%	726
	PMMRL	**23.33%**	**0.65**	**1.18**	**0.4614**	**−14.57%**	**432**
600398.SH	B&H	−20.33%	−0.08	−0.12	−0.1320	−45.15%	578
	DEMA	−34.28%	−0.47	−0.68	−0.2424	−44.26%	1265
	FDDR	−20.41%	−0.02	−0.03	−0.1326	−45.15%	578
	PTRL	7.34%	**0.42**	**0.80**	**0.3618**	**−5.34%**	**378**
	GDQN	−6.79%	−0.46	−0.51	−0.1711	−11.01%	518
	PMMRL	**14.02%**	0.35	0.53	0.2302	−15.69%	703

nize the current market trend and still generate trade signals, which causes a severe loss, as shown in the red box of Fig. 6(b). But the PMMRL agent algorithm doesn't make any transactions in this downward trend, which avoids the loss. Further, there are many profit opportunities during the upward trend, the GDQN agent doesn't take any transaction or position, as shown in the red box of Fig. 6(c). But the PMMRL agent gains a lot of profit during this period from the curve of the total asset (colored in yellow). Thus, based on the above analysis, we find PMMRL algorithm can find almost all valid trading signals and make a profit.

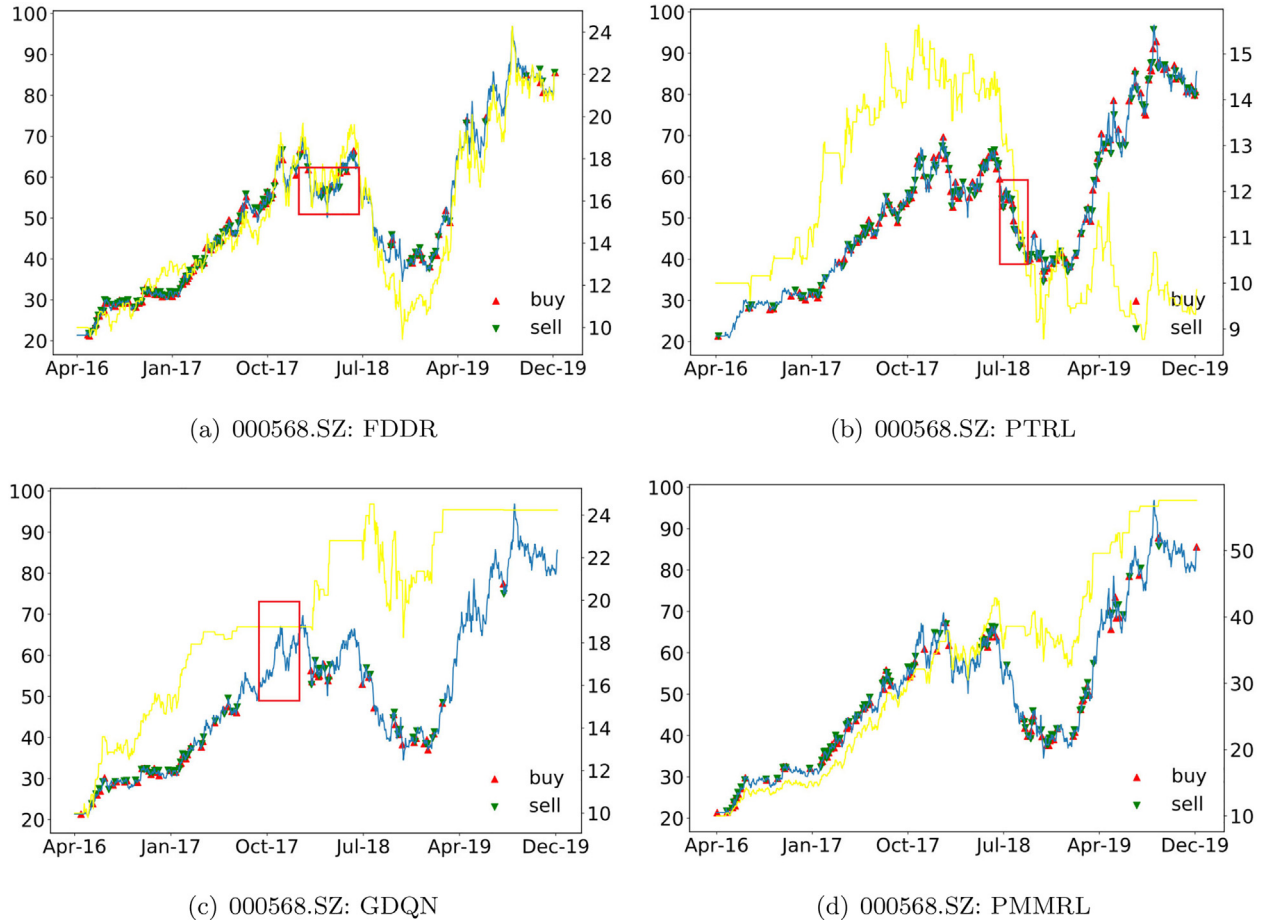
Gaining profit in the bull market is very easy, but gaining stable profit or stopping loss in the bear or consolidation market is very difficult. According to these results, we can conclude the proposed PMMRL algorithm can take a steady profit under any market state.

4.3.3. Ablation experiments

In this part, we conduct several ablation experiments to explain the importance of the 'parallel multi-module' structure used to extract features. The following two experiments are implemented on the same stock dataset.

- **FC-RL** is a model which only use FC module to extract all features, that is, $\mathscr{S} = FC(S_1, S_2, S_3)$. The corresponding state input of the agent is $\mathscr{S}^{new} = \{\mathscr{S}, h_t, b_t\}$. The FC module structure is the same as the previous FC module in PMMRL, and other settings are also the same as the corresponding setting of the PMMRL algorithm.
- **LSTM-RL** is a model which extracts all features by LSTM and FC modules in series, that is, $\mathscr{S} = FC(LSTM(S_1, S_2, S_3))$. The corresponding state input of the agent is $\mathscr{S}^{new} = \{\mathscr{S}, h_t, b_t\}$. Here, LSTM and FC modules have the same structure as the modules of PMMRL, and other settings are also the same as the PMMRL.

These results are shown in Table 3. From the results of FC-RL and LSTM-RL algorithms, we can see the cumulative return has been improved a little when adding the LSTM module into the



(a) 000568.SZ: FDDR (b) 000568.SZ: PTRL

(c) 000568.SZ: GDQN (d) 000568.SZ: PMMRL

Fig. 6. The comparison of trading points between different models. The horizontal axis is date. The left vertical axis is close price, and the unit is yuan. The right vertical axis is total asset value, and the unit is 10000 yuan. The close prices and total asset values are represented by blue and yellow curves, respectively.

algorithm to encode the related feature, which explains the validity of the LSTM module. Further, comparing the performance of LSTM-RL with PMMRL algorithms, we find that cumulative return, Sharpe ratio have been significantly improved. The significant difference between the two algorithms is feature extraction, that is, PMMRL algorithm encodes different features by different modules and concats them in parallel; LSTM-RL algorithm encodes the features by the two modules connected in sequence. From these results, we can conclude that it is essential to extract feature using appropriate module according to the characteristics of the data. It thoroughly explains the effectiveness of the 'parallel multi-module' mechanism.

5. Conclusion and future work

Inspired by the classical CAPM, the expected return of an investment is not only influenced by the price of the traded stock, but also influenced by the fundamental data of the issuing company and the price sequences of the related stocks. Considering the characteristics of these data, we propose a novel DRL algorithm, called *Parallel Multi-Module Reinforcement Learning* (PMMRL) algorithm, which extracts features from the overall financial environment by adopting FC and LSTM modules in parallel. Here, FC module extracts the current state from market data and the fundamental data of the traded stock. Meanwhile, LSTM module extracts long-term market features from the historical price sequences of the traded stock and the related stocks. In this way, the agent considers the current state and the long-term historical trend when taking action. Encoding features in this parallel way is superior to the other traditional algorithms. Experimental results on the China Stock Market demonstrate the proposed PMMRL algorithm can take more profit and undertake less risk under any market state.

In this paper, we build an automatic trading system based on the RL algorithm, which is better than the other existing models. But there are some limitations to this algorithm. Firstly, the fundamental data is essential for the agent to recognize the current state, but it is impossible to get similar information for future or digital currency. Hence, this algorithm can only be used for trading stock at present. Secondly, the proposed algorithm has achieved good performance in trading a single stock, but how to use this algorithm in portfolio management needs further research. Besides, risk management needs to be considered for a complete trading system. Therefore, building an investment strategy for portfolio management and controlling the trading system's risk based on RL agents are our future work.

CRediT authorship contribution statement

Cong Ma: Data curation, Software, Methodology, Visualization, Writing - original draft, Writing - review & editing. **Jiangshe Zhang:** Supervision, Resources, Writing - review & editing. **Junmin Liu:** Writing - review & editing. **Lizhen Ji:** Writing - review & editing. **Fei Gao:** Data curation.

Table 3
Ablation experiment results of PMMRL algorithm.

Stock codes	Methods	Cumulative return	Sharpe ratio	Sortino ratio	Calmar ratio	MDD	Longest DD days
000568.SZ	FC-RL	61.01%	0.79	1.22	0.5004	−27.46%	713
	LSTM-RL	81.01%	1.61	**5.57**	**4.7137**	**−3.69%**	**19**
	PMMRL	**475.89%**	**1.85**	3.11	2.4753	−24.46%	245
000768.SZ	FC-RL	48.76%	0.56	0.81	0.3594	−36.50%	330
	LSTM-RL	−16.93%	−0.02	−0.03	−0.1118	−50.05%	993
	PMMRL	**63.04%**	**0.72**	**1.05**	**0.6466**	**−25.34%**	**266**
000860.SZ	FC-RL	347.01%	1.36	2.22	1.4613	−34.15%	**223**
	LSTM-RL	374.00%	1.34	2.24	1.5306	−34.17%	643
	PMMRL	**371.78%**	**1.40**	**2.36**	**1.5343**	−33.96%	351
002032.SZ	FC-RL	247.48%	1.17	1.96	1.8493	−21.65%	233
	LSTM-RL	314.76%	1.77	3.31	2.8388	−16.52%	**147**
	PMMRL	**533.42%**	**1.86**	**3.38**	**4.0783**	**−15.87%**	155
600000.SH	FC-RL	14.79%	0.30	0.46	0.1165	−32.60%	896
	LSTM-RL	13.77%	0.29	0.44	0.1103	−32.18%	819
	PMMRL	**97.89%**	**1.46**	**2.47**	**1.9494**	**−10.40%**	**196**
600036.SH	FC-RL	34.34%	1.54	**3.36**	**2.4274**	**−3.42%**	**57**
	LSTM-RL	252.16%	1.57	2.55	1.4795	-27.40%	415
	PMMRL	**365.95%**	**2.03**	3.35	1.8424	−28.01%	381
600498.SH	FC-RL	24.72%	0.35	0.50	0.1576	−39.10%	777
	LSTM-RL	216.45%	1.12	1.64	0.9826	−37.22%	774
	PMMRL	**394.39%**	**1.62**	**2.53**	**2.5781**	**−20.98%**	**111**
601628.SH	FC-RL	48.62%	0.62	0.96	0.3785	**−37.41%**	711
	LSTM-RL	48.01%	0.60	0.94	0.3377	−41.47%	714
	PMMRL	**69.66%**	**0.78**	**1.23**	**0.4671**	−41.37%	**694**

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work is supported by the National Key Research and Development Program of China (No. 2020AAA0105601), in part by the National Natural Science Foundation of China (Nos. 61976174, 61877049), and part by the program of China Scholarships Council (No. 201906280201).

Appendix A. Technical indicators

There are several technical indicators used in this paper to extract the features of price and volume.

- **Moving average (MA)** is widely used to smooth out the price in finance, which can be expressed as

$$MA_t^k = \frac{1}{k}\sum_{i=t-k+1}^{t} p_i,$$

where t is the current time, k is the look-back period, and p_i is the close price at time i.
- **Moving average convergence divergence (MACD)** is a trend-following momentum indicator that reveals the changes in the strength, direction, momentum, and duration of a trend in one stock price [35]. It can be described as the following formulas

$$EMA_t^k(p_t) = \frac{2}{k+1}\left[p_t - EMA_{t-1}^k(p_{t-1})\right] + EMA_{t-1}^k(p_{t-1}),$$

$$MACD_t = EMA_t^{k_1}(p_t) - EMA_t^{k_2}(p_t),$$

where $EMA_1^k = p_1, EMA_t^k(p_t)$ is the exponential moving average of close price with time window k at time t, k_1 and k_2 are different look-back periods, and $k_1 > k_2$.
- **Relative strength index (RSI)** depicts the current and historical strength or weakness of a stock based on the close price of a recent trading period [36], which is defined as

$$SMMA_t(U_t, k) = EMA_{t-1}^k(U_{t-1}) + \frac{1}{k}\left(p_t - EMA_{t-1}^k(U_{t-1})\right),$$

$$RS_t = \frac{SMMA_t(U_t, k)}{SMMA_t(D_t, k)},$$

$$RSI_t = 100 - \frac{100}{1 + RS_t},$$

where $SMMA_t$ is the smoothed moving average at time t, RS_t is the relative strength at time $t, U_t = max(p_t - p_{t-1}, 0), D_t = max(0, p_{t-1} - p_t)$ are upward change and downward change, respectively.
- **On balance volume (OBV)** is generally used to confirm price moves [37], and can be expressed as

$$OBV_t = OBV_{t-1} + sgn(p_t - p_{t-1}) * V_t,$$

where $sgn(\cdot)$ is the sign function, and V_t is the volume at time point t.

Appendix B. Evaluation criteria

Here, we introduce several evaluation criteria used in this paper.

- **Cumulative return ratio** is the most direct indicator to measure profit, which can be calculated as follows

$$cumulative\ return\ ratio = \frac{TAV_T - TAV_1}{TAV_1} \times 100\%,$$

Table C.5
Stock information.

Stock traded	Sector/Stock Name	Relative stocks	Training period	Test period
000559.SZ	Automotive manufacturing/ Wanxiang Qianchao	600303.SH, 600418.SH, 600166.SH, 000700.SZ	2005/01/01–2016/04/19	2016/04/20–2019/12/31
000630.SZ	Nonferrous metals/ Tongling Nonferrous Metals	000878.SZ, 600362.SH, 000060.SZ, 000960.SZ	2005/01/01–2016/04/19	2016/04/20–2019/12/31
000785.SZ	Retail/ Easyhome New Retail Group	600824.SH, 600861.SH, 600814.SH, 000759.SZ	2005/01/01–2016/04/19	2016/04/20–2019/12/31
000799.SZ	Chinese liquor/ Jiugui Liquor	000858.SZ, 600809.SH, 000568.SZ, 000860.SZ	2005/01/01–2016/04/19	2016/04/20–2019/12/31
000895.SZ	Food/ Henan Shuanghui	002557.SZ, 002311.SZ, 300138.SZ, 002385.SZ	2012/01/01–2017/12/11	2017/12/12–2019/12/31
600104.SH	Automotive manufacturing/ SAIC Motor	000625.SZ, 600741.SH, 000800.SZ, 600418.SH	2005/01/01–2016/04/19	2016/04/20–2019/12/31
600221.SH	Civil aviation/ Hainan Airlines Holding	600029.SH, 601111.SH, 600115.SH, 000099.SZ	2007/01/01–2016/09/30	2016/10/10–2019/12/31
600398.SH	Clothing/ Hla Corp	600177.SH, 600400.SH, 002003.SZ, 600655.SH	2005/01/01–2016/04/19	2016/04/20–2019/12/31

where TAV_T is the total asset value at the end time T, TAV_1 is the initial asset value.

- **Sharp ratio** is the average return earned over the risk-free rate per unit of volatility or total risk. The formula is

$$SharpeRatio = \frac{R_p - R_f}{\sigma_p},$$

where R_p is the return of the trading, R_f is risk-free rate, and σ_p is standard deviation of the excess return. The higher the Sharpe ratio, the better the investment strategy.

- **Sortino ratio** is a modification of Sharpe ratio but only penalizes downside volatility, which can be expressed as

$$SortinoRatio = \frac{R_p - R_f}{\sigma_d},$$

where σ_d is the standard deviation of the downside. Also, the greater this value, the better the investment strategy.

- **Maximum drawdown (MDD)** measures the size of the largest loss, which looks for the greatest movement from a high point to a low point. It can be calculated as follows

$$MDD = \max_{0 \leqslant t_1 \leqslant t_2 \leqslant T} \frac{R_{t_1} - R_{t_2}}{R_{t_1}},$$

where T is the length of trading period, R_{t_1}, R_{t_2} are the cumulative return at the time t_1 and t_2, respectively.

- **Longest drawdown (DD) days** is the time span from the cumulative return R_{t_1} to R_{t_2}, that is, $t_2 - t_1$, where t_1 and t_2 are the time points defined in MDD. The longer this span, the greater the risk of the investment strategy.
- **Calmar ratio** evaluates the performance of the investment relative to its risk. It can be calculated as follows

$$CalmarRatio = \frac{r_p}{MDD},$$

where r_p is the average annual rate of return. A higher value means a better investment.

Table C.4
The list of key notations.

Notations	Description
p_t	close price at time t
h_t	amount of the holding at time t
b_t	unrealized profit and loss at time t
r_p	average annual rate of return
k	look-back period
σ_p	standard deviation of the excess return
σ_d	standard deviation of the downside
R_t	cumulative return at time t
R_p	expeted return of the trading
R_f	risk-free rate
TAV_t	total asset value at time t
MA_t	moving average of close price at time t
$MACD_t$	moving average convergence divergence at time t
RSI_t	relative strength index at time t
OBV_t	on balance volume at time t

Appendix C. Key notations

Here, we list all necessary notations of this paper in Table C.4.

Appendix D. Experiment Results

Table C.5 introduces the information of more stocks used in the supplementary experiment. And Table C.6 shows the corresponding experiment results to compare the proposed algorithm and several baselines in China Stock Market.

References

[1] E.A. Gerlein, M. McGinnity, A. Belatreche, S. Coleman, Evaluating machine learning classification for financial trading: an empirical approach, Expert Syst. Appl. 54 (2016) 193–207.

[2] F.D. Paiva, R.T.N. Cardoso, G.P. Hanaoka, W.M. Duarte, Decision-making for financial trading: a fusion approach of machine learning and portfolio selection, Expert Syst. Appl. 115 (2019) 635–655.

[3] J. De Spiegeleer, D.B. Madan, S. Reyners, W. Schoutens, Machine learning for quantitative finance: fast derivative pricing, hedging and fitting, Quantitative Finance 18 (10) (2018) 1635–1643.

[4] A. Booth, E. Gerding, F. Mcgroarty, Automated trading with performance weighted random forests and seasonality, Expert Syst. Appl. 41 (8) (2014) 3651–3661.

[5] K.O. Nti, A. Adekoya, B. Weyori, Random forest based feature selection of macroeconomic variables for stock market prediction, Am. J. Appl. Sci. 16 (7) (2019) 200–212.

[6] W. Huang, Y. Nakamori, S.-Y. Wang, Forecasting stock market movement direction with support vector machine, Comput. Oper. Res. 32 (10) (2005) 2513–2522.

[7] Y. Chen, Y. Hao, Integrating principle component analysis and weighted support vector machine for stock trading signals prediction, Neurocomputing 321 (2018) 381–402.

[8] Q. Wang, W. Xu, X. Huang, K. Yang, Enhancing intraday stock price manipulation detection by leveraging recurrent neural networks with ensemble learning, Neurocomputing 347 (2019) 46–58.

[9] C. Chen, P. Zhang, Y. Liu, J. Liu, Financial quantitative investment using convolutional neural network and deep learning technology, Neurocomputing 390 (2020) 384–390.

[10] D. Silver, A. Huang, C.J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al., Mastering the game of go with deep neural networks and tree search, Nature 529 (7587) (2016) 484–489.

[11] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533.

[12] Z. Li, S. Xue, W. Lin, M. Tong, Training a robust reinforcement learning controller for the uncertain system based on policy gradient method, Neurocomputing 316 (2018) 313–321.

[13] Z. Jiang, J. Liang, Cryptocurrency portfolio management with deep reinforcement learning, in: 2017 Intelligent Systems Conference (IntelliSys), IEEE, 2017, pp. 905–913..

[14] C. Li, X. Wei, Y. Zhao, X. Geng, An effective maximum entropy exploration approach for deceptive game in reinforcement learning, Neurocomputing 403 (2020) 98–108.
[15] C. Ma, Z. Li, D. Lin, J. Zhang, Parallel multi-environment shaping algorithm for complex multi-step task, Neurocomputing 402 (2020) 323–335.
[16] R. Neuneier, Optimal asset allocation using adaptive dynamic programming, in: Advances in Neural Information Processing Systems, 1996, pp. 952–958..
[17] J. Moody, L. Wu, Y. Liao, M. Saffell, Performance functions and reinforcement learning for trading systems and portfolios, J. Forecast. 17 (5–6) (1998) 441–470.
[18] M.A. Dempster, V. Leemans, An automated fx trading system using adaptive reinforcement learning, Expert Syst. Appl. 30 (3) (2006) 543–552.
[19] Y. Deng, F. Bao, Y. Kong, Z. Ren, Q. Dai, Deep direct reinforcement learning for financial signal representation and trading, IEEE Trans. Neural Netw. Learn. Syst. 28 (3) (2016) 653–664.
[20] F. Rundo, Deep LSTM with reinforcement learning layer for financial trend prediction in FX high frequency trading systems, Appl. Sci. 9 (20) (2019) 4460.
[21] X. Xing, J.S. Howe, The empirical relationship between risk and return: evidence from the uk stock market, Int. Rev. Financ. Anal. 12 (3) (2003) 329–346.
[22] W.F. Sharpe, Capital asset prices: a theory of market equilibrium under conditions of risk, J. Finance 19 (3) (1964) 425–442.
[23] T.G. Bali, S.J. Brown, M.O. Caglayan, Systematic risk and the cross section of hedge fund returns, J. Financ. Econ. 106 (1) (2012) 114–131.
[24] A. Beja, On systematic and unsystematic components of financial risk, J. Finance 27 (1) (1972) 37–45.
[25] L.A. Cox, G.L. Griepentrog, Systematic risk, unsystematic risk, and property-liability rate regulation, J. Risk Insurance (1988) 606–627.
[26] E.F. Fama, K.R. French, Common risk factors in the returns on stocks and bonds, J. Financ. Econ. 33 (1993) 3–56.
[27] H. v. Hasselt, A. Guez, D. Silver, Deep reinforcement learning with Double Q-Learning, in: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, AAAI Press, 2016, p. 2094–2100..
[28] C. J. C. H. Watkins, Learning from delayed rewards., Ph.D. thesis, University of Cambridge (1989)..
[29] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., Tensorflow: A system for large-scale machine learning, in: 12th USENIX Symposium on Operating Systems Design and Implementation OSDI 16, 2016, pp. 265–283..
[30] Jointquant,https://www.joinquant.com..
[31] B.G. Malkiel, A random walk down Wall Street: including a life-cycle guide to personal investing, WW Norton & Company (1999).
[32] P.G. Mulloy, Smoothing data with faster moving averages, Stocks Commodities 12 (1) (1994) 11–19.
[33] K.S. Zarkias, N. Passalis, A. Tsantekidis, A. Tefas, Deep reinforcement learning for financial trading using price trailing, in: ICASSP 2019–2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2019, pp. 3067–3071.
[34] X. Wu, H. Chen, J. Wang, L. Troiano, V. Loia, H. Fujita, Adaptive stock trading strategies with deep reinforcement learning methods, Inf. Sci. 538 (2020) 142–158.
[35] G. Appel, Technical analysis: power tools for active investors, FT Press (2005).
[36] J.W. Wilder, New concepts in technical trading systems, Trend Res. (1978).
[37] J.E. Granville, Granville's New Strategy of Daily Stock Market Timing for Maximum Profit, Prentice-Hall, 1976.

Cong Ma received the B.S. degree in Mathematics and Applied Mathematics from Shanxi University, Taiyuan, China, in 2015. She is currently pursuing her Ph.D. degree in the School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an, China. Her research interests are focused on deep learning and reinforcement learning, especially their application in finance.

Jiangshe Zhang received the B.S., M.S., and Ph.D. degrees in Computational Mathematics from Xi'an Jiaotong University, Xi'an, China, in 1984, 1987, and 1993, respectively. He is currently the Director of the Institute of Machine Learning and Statistical Decision Making, Xi'an Jiaotong University, where he is also a Professor with the Department of Statistics. He is also the Vice-President of the Xi'an International Academy for Mathematics and Mathematical Technology, Xi'an. He has authored and co-authored one monograph and over 100 journal papers. His current research interests include statistical computing, deep learning, cognitive representation, and statistical decision making. Dr. Zhang received the National Natural Science Award of China (Third Place) in 2007 and the First Prize in Natural Science from the Ministry of Education of China in 2007. He served as the President of the Shaanxi Mathematical Society and the Executive Director of the China Mathematical Society.

Junmin Liu received the M.S. degree in Computational Mathematics from Ningxia University, Yinchuan, China, in 2009, and the Ph.D. degree in Applied Mathematics from Xi'an Jiaotong University, Xi'an, China, in 2013. He is currently an Associate Professor with the School of Mathematics and Statistics, Xi'an Jiaotong University. His current research interests include hyperspectral unmixing, remotely sensed image fusion, and deep learning.

Lizhen Ji received the B.S. degree in Mathematics and Applied Mathematics from Northwest University, Xi'an, China, in 2014. She received her M.S. degree in Statistics from UC Davis, in 2016. She is currently pursuing her Ph. D. degree in the School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an, China. Her research interests are in the area of deep learning and neural network compression.

Fei Gao received the B.S. degree in Financial Mathematics and Statistics from Northwest University, Xi'an, China, in 2016. She is currently pursuing the Ph.D. degree in the School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an, China. Her research interests include deep learning and reinforcement learning, especially their application in finance.