Applied Soft Computing 107 (2021) 107320

Contents lists available at ScienceDirect

Applied Soft Computing

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

Stock trading rule discovery with double deep Q-network

Yong Shi [a], Wei Li [b], Luyao Zhu [b,*], Kun Guo [a], Erik Cambria [b]

[a] School of Economics and Management, University of Chinese Academy of Sciences, Beijing 100190, China
[b] School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798, Singapore

ARTICLE INFO

Article history:
Received 9 June 2020
Received in revised form 5 February 2021
Accepted 7 March 2021
Available online 23 March 2021

Keywords:
Deep reinforcement learning
Double DQN
Stock price trend forecasting
Stock trading

ABSTRACT

Stock market serves as an important indicator of today's economy. Predicting the price fluctuation of stocks and acquiring the maximum gains has been the main concern of investors. In recent years, deep learning models are widely applied to stock market prediction and have achieved good performances. However, the majority of these deep learning based models belong to supervised learning methods and are not capable of dealing with long-term targets. Therefore, in this paper we proposed a deep reinforcement learning based stock market trading model, which is suitable for predicting stock price fluctuation and stock transactions. We carefully devise the reward function and deep learning based policy network, which enables the model to capture the hidden dependencies and latent dynamics in the stock data. In order to evaluate the superiority of the proposed model, stock price trend forecasting and transaction is conducted on randomly selected stocks and stock market indices. Experiment results demonstrate that our model outperforms baseline methods on several indicators.

1. Introduction

Stock market forecasting has been the main research focus of researchers for a long time. In the past years, a lot of forecasting models have been created to assist investors to make better decisions and obtain maximum gains. On the other hand, it can also help financial institution to control risk and reduce loss. Stock market acts as an indicator of economy nowadays. However, due to that stock market is an intricate nonlinear dynamic system with uncertain stochastic disturbance, the price fluctuation, to some extent, is hard to forecast. Besides, complicated market rules make it even impractical for traditional regression models to handle these issues. Therefore, it becomes a core problem to design effective algorithms to extract useful information from massive structured data and further make accurate prediction and investment policy for financial time series.

The time series forecasting problems are usually classified as long-term, medium-term and short-term forecasting. The short-term forecasting for stock price usually contains prediction for a few seconds, minutes, days, weeks and months. In this work, the stock price is aimed at short-term prediction for a few months via training on daily stock information. The existing method for stock price prediction includes: (1) Fundamental Analysis and (2) Technical Analysis. The fundamental analysis evaluates the share value through financial statement analysis, which is more suitable for long-term forecasting. While technical analysis focuses on the historical stock price time series and it is more suited to short-term forecasting. Little access to high-frequency financial statement data and the goal to make short-term prediction lead to the application of technical analysis method in this work. As is known, the technical analysis methods consist of two model families: (1) linear models and (2) non-linear models. However, the linear models define linear equations for univariate time series, thus they cannot account for the inter dependencies among the various factors in the data. Recently, some non-linear models, like support vector machine (SVM) and artificial neural networks (ANNs), solve the aforementioned problem and are proved to outperform the linear models in stock price prediction [1–4]. There has been a great increase of interest in ANNs [5] and reinforcement learning (RL), due to their success in natural language processing (NLP) and computer vision [6–8]. Albeit, the application of deep reinforcement learning (DRL) on stock price forecasting is still limited.

In this work, we propose a double deep Q-network (Double DQN) model, where a novel deep learning model is designed as a policy network, for the prediction of stock price trend and stock trading.[1] In the policy network, multiple Convolutional Neural Network (CNN) layers with different kernel size are employed to extract the latent relations among the transaction data on different transaction dates. Besides, we evaluated the performance of models using two designed reward functions, accrual basis and cash basis. The experiment results also show the stability of the

* Corresponding author.
E-mail address: luyao001@e.ntu.edu.sg (L. Zhu).

[1] Our code is available at https://github.com/Cyn7hia/Double-DQN-stock-trading-rule.

https://doi.org/10.1016/j.asoc.2021.107320
1568-4946/

propose model by comparing the different architectures of the policy network.

The main contributions of this work are shown as follows:

1. Application of end-to-end Double DQN model to financial time series analysis problem, taking into account the transaction cost in both training and testing phases;
2. Well-designed CNN to capture the hidden dependencies and latent dynamics in the stock data;
3. Proposing two kinds of reward functions in the Double DQN for training and testing;

The rest of this paper is structured as follows: Section 2 introduces related works on deep learning, RL, DQN, and stock price forecasting; Section 3 elaborates the stock market environment for DRL; the mechanism of the proposed Double DQN model is explained in Section 4; the data description, experiment and evaluation are described in Section 5; finally, Section 6 provides some concluding remarks.

2. Related work

2.1. Stock price forecasting models

According to paper [9], the majority of the traditional models/techniques for stock price prediction includes the Autoregressive (AR) [10], Moving Average (MA) [11], Autoregressive Moving Average (ARMA) [12,13], Autoregressive Integrated Moving Average (ARIMA) [14,15], Random Walk (RW) [16,17], linear and non-linear regressive models [18], Generalized Autoregressive Conditional Heteroskedasticity (GARCH) family models [16], Buying and Hold (B&H) strategy [19], and Stochastic Volatility (SV) models [20]. The extensions of these models mentioned above, applied to stock price prediction task, include FIGARCH model [21], ARRIMA model [22], ARFIMA-FIGARCH model [23], Vector Autoregressive (VAR) models [24] and the Exponentially-weighted (EWMA) model [25]. Large amounts of research experiment results indicate that the GARCH family models outperform other traditional methods such as AR, ARIMA, RW, B&H etc. However, the latent dependencies and dynamics existing in the data could be complicated and the traditional models cannot identify them precisely. Besides, designing of the regression model might be time-consuming since it comes to a relatively satisfactory model through trial and error.

With the development and the resurgence of interest in artificial intelligence (AI), much attention has been drawn to the application of machine learning methods, such as SVM, ANNs, K-nearest Neighbor (KNN), etc., on stock price prediction. The first kind of AI application on financial time series is under the Efficient Market Hypothesis whereby share prices reflect all information and consistent alpha generation is impossible. [26] is the first research to establish a financial time series model through ANNs. Recently, Chiang et al. [27] combined the ANNs and particle swarm optimization to simulate the trading of World 22 stock market indices. Zhong and Enke [28] predicted the market direction of US S&P 500 ETF (SPY) through dimension reduction and ANN. In 2017, Chong et al. [29] proposed a systematic analysis of the Korea KOSPI stock market and predicted the return via data representation and deep neural networks. Selvin et al. [30] predicted the price of National Stock Exchange by three different deep learning approaches, Long Short-Term Memory (LSTM) model, Recurrent Neural Network (RNN) and CNN separately, and compared the experiment results with ARIMA model. The results showed that CNN surpassed other methods. Besides, Ballings et al. [31] and Gerlein et al. [32] utilized SVM, KNN, ANNs and decision trees and gained good performances. Zbikowski [33], Kumar et al. [34] utilized SVM based systems with Technical Analysis indicators to create stock trading strategy and forecast trend in financial market. At the same time, some researches [35–38] used support vector regression (SVR), a regression method based on SVM, to obtain good results in stock price prediction.

In addition, there are some forecasting researches employing the theory of market irrationality which means there are some speculative bubbles in the financial market. Under this theory, the price cannot reflect all the information in the market. This calls for effective NLP method to extract exterior information, such as investor sentiments, from the text and incorporate these information with financial data for subsequent forecasting, which is called natural language based financial forecasting [39]. Ding et al. [40] leveraged CNN for sentiment analysis to extract event information for event-driven stock prediction. Xing et al. [41] used ECM-LSTM, which is based on evolving clustering method and Sentic LSTM [42], to formalize sentiment information into market views for intelligent asset allocation via a Bayesian approach. Picasso et al. [43] combined the fundamental and technical analysis methods to forecast market trend by means of applying machine learning techniques to time series prediction and sentiment analysis.

2.2. Reinforcement Learning

Reinforcement Learning (RL) has drawn much attention since the last century [44–46]. The algorithm is designed to learn an intelligent agent through trial and error, where the output of the designed environment is a reward. The task of the agent is to maximize the discounted reward from the environment per action [47].

An intelligent agent can be obtained through value iteration and policy iteration. Among these, Q-learning methods [48,49] and actor–critic methods [50] are two famous policy iteration methods in RL. As deep learning methods continues to mature, some researches focus on employing deep neural networks to learn Q matrix. In 2015, Mnih et al. proposed a DQN model to learn long-term control policies with smattering prior knowledge [51]. DQN exploited the replay algorithm [52–54] and successfully integrated RL with deep neural networks. This reduced the computation complexity and improved the performance of RL agents because of the rich representations provided by deep neural networks [55]. However, some researchers believed that the integration of online RL with deep neural networks was unstable [51,56–59]. Hasselt et al. [58] proposed a Double DQN algorithm to reduce the overestimation. Mnih et al. [55] put up a parallel RL paradigm, which asynchronously executed multiple agents on multiple instances of the environment in parallel, to deal with the non-stationary problem. Besides, the algorithm was compatible with various RL algorithms, including on-policy ones such as actor–critic methods, Sarsa, n-step methods, and off-policy ones like Q-learning. Moreover, DQN only dealt with discrete and low-dimensional action spaces, instead of continuous domains [60]. And adapting the DQN to continuous action domain by simply discretization would cause the curse of dimensionality. Thus, Lillicrap et al. [60] presented an actor–critic, model-free algorithm named deep deterministic policy gradient, to learn polices "end-to-end" and achieved a good performance in continuous action domain tasks which is competitive with results found by other algorithms fully fed with the dynamics of the domain and the derivatives. By now, the DQN has been applied to a broad spectrum of tasks, such as play Atari 2600 games [57,61], natural sentence regeneration [62], language understanding [63], classic simulated physics tasks [60], etc.

These are also some application of RL on financial time series forecasting. Lee [64] used RL algorithm TD(0) to predict stock price in Korean stock market. Rutkauskas and Ramanauskas [65]

established heterogeneous agents whose individual actions were driven by Q-learning algorithm in a stock market model to analyze the market efficiency and self-regulation abilities, and determinants emergent properties of the financial market. Nevmyvaka et al. [66] utilized the RL to optimize trade execution in financial markets based on large-scale NASDAQ microstructure datasets. Li et al. [67] proposed a RL scheme for short-term stock price movements prediction by actor–critic and actor-only RL methods, respectively. Tan et al. [68] used the RL paradigm to determine the periods for moving averages and momentum and used ANFIS-RL to deal with the delay in the predicted cycle. Nevertheless, the number of research work on application of DRL is still limited.

In summary, the classic stock price forecasting method such as ARIMA, GARCH, lacks the improvement potential in accuracy since it cannot deal with more complicated dependency between data and is time-consuming to gain an optimal model through trial and error; the machine learning based methods show better performances than classic regression based models, but these methods pay more attention on accuracy instead of revenue or return rate; Meanwhile, the recent research progress of DRL models shows that is possible to improve the accuracy of stock price trend prediction through DRL with well-designed deep neural networks. This is the inspiration of our end-to-end Double DQN for stock price trend prediction and transaction.

3. Stock market environment

Unlike other application areas of DRL such as games and robots, the environment designed for stock market is more sophisticated, since there are a lot of factors influencing the stock transaction, pricing, etc. in the market. The environment settings may influence the model selection and experimental results. Besides, the information contained in the environment need to be introduced and carefully selected to formulate part of the state in DRL, which will be transited to the RL agent. To this end, a detailed stock market environment definition and corresponding feature selection and extraction procedures are introduced in this section.

3.1. Problem definition

In this work, the simulated stock market is populated by a large number of heterogeneous investors. Although the stock market is more like an imperfectly competitive market, it is supposed that the DRL agent (investors) is not strong enough to influence the price of the stock. Besides, all the investors differ in their financial endowments, risk preference and expectations on stock price. In this case, the artificial DRL investor in this work would not influence the pricing of the selected stocks through transaction. And borrowing is allowed. Initially the DRL investor is endowed with adequate cash holdings, and subsequently it may submit an order to long/buy or short/sell one unit of stock in every trading. It is assumed that the stock transactions are subject to the T+1 settlement date. All the stocks are traded at the closing price in the period.

The following notations are adopted for the variables in this study.

a. Notations for stock market environment

t: transaction date;
Op_t: opening price at time step t;
Cl_t: closing price at time step t;
Hi_t: highest traded price at time step t;
Lo_t: lowest traded price at time step t;
V_t: transaction volume at time step t;
P_t: position at time step t after carrying out some strategy during past period;
d_t: duration of an invariant action in a steady strategy by the time step t;
v_t: volume of single investor under some strategy at time step t;
r_t^j: return of stock j at time step t;
σ_t: volatility of the stock price at time step t;
c: trading cost;
γ: discount rate;
w: window size at each state;

b. Notations for reinforcement learning:

A_t: action set at time step t;
a_t^i: action i in set A_t at time step t
S_t: state at time step t
R_t: reward at time step t;

c. Notations for policy network architecture:

W_{DLN}: weights of deep learning network in policy network;
W_{Target}: weights of target network;
$\pi\left(\boldsymbol{a}|S; W_{DLN}\right)$: policy/distribution over actions $\boldsymbol{a}$ given states S in DQN;
$Q\left(S_t, a_t; W_{DLN}\right)$: action-value function with weights W_{DLN}, which is the expected reward starting from state S_t, taking action a_t;
$Q'\left(S_t, a_t; W_{Target}\right)$: target network with weights W_{Target}, which is the expected reward starting from state S_t, taking action a_t;
$Y_t^{TargetDQN}$: target Q value in target network.

3.2. Feature selection and extraction

In many stock price trend prediction models, feature selection and extraction is a pivotal procedure, since it will significantly influence the performance of the model. Thus, in many researches, this process is emphasized and some stock technical analysis indicators from explicit feature extraction, like exponential moving average, momentum index, relative strength, etc. are introduced into the model. However, this requires large amounts of computation of indicators before the data are fed into the prediction model; Besides, feature selection is necessary for regression model when there are plenty of calculated technical indicators, since the multicollinearity problem may exist. Different from these researches, an end-to-end model is designed in this work, where the feature selection is simplified according to the Efficient Market Hypothesis and no other technical indicators are explicitly extracted except the log growth rate of the input; The latent dependency among the stock information data could be captured by the well-designed deep learning network.

According to the Efficient Market Hypothesis, the stock price information is sufficient to reflect all the information in the stock market; thus, there is no need to introduce other financial indicators from the financial statements of selected stocks. This work focuses on forecasting the trend of the selected stock price (whether increase or decrease). In this case, it is more rational to use the change of the features over time instead of the absolute value of it. I_{t+1} is the input fed to the whole model at time step $t+1$. The input matrix I_{t+1} is given by

$$I_{t+1} = (Op_t, Cl_t, Hi_t, Lo_t, V_t), \tag{1}$$

where $Op_t, Cl_t, Hi_t, Lo_t, V_t$ are the opening price, closing price, highest price, lowest price, and transaction volume of the selected stock at time step t, respectively. In this work, it is assumed that the price of the stock changes continuously, thereby the log return is introduced to represent the price change of the stock. The corresponding extracted feature is the change rate from the input I_t to I_{t+1} between two adjacent transaction dates, which can be calculated by

$$\nabla Op_{t-1} = lnOp_t - lnOp_{t-1}, \tag{2}$$

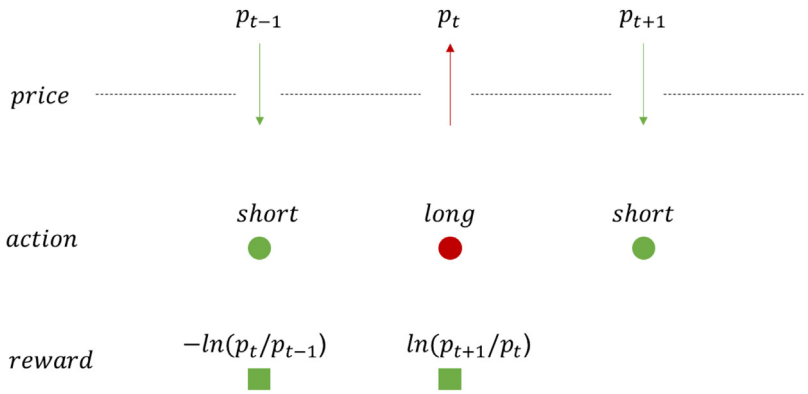

Fig. 1. Feedbacks in RL for stock price trend prediction.

$$\nabla Cl_{t-1} = lnCl_t - lnCl_{t-1}, \tag{3}$$

$$\nabla Hi_{t-1} = lnHi_t - lnHi_{t-1}, \tag{4}$$

$$\nabla Lo_{t-1} = lnLo_t - lnLo_{t-1}, \tag{5}$$

$$\nabla V_{t-1} = lnLo_t - lnLo_{t-1}. \tag{6}$$

$$\nabla I_{t+1} = (\nabla Op_{t-1}, \nabla Cl_{t-1}, \nabla Hi_{t-1}, \nabla Lo_{t-1}, \nabla V_{t-1}). \tag{7}$$

Then, the normalized features extracted from the input is shown as Eq. (7). Besides, the position P_t at time t is also selected as a feature, which could be obtained from the strategy carried out during the past n days.

$$P_t = \sum_{i=1}^{n} \omega_{t-i} v_{t-i}, \tag{8}$$

where

$$\omega_{t-i} = \begin{cases} -1, \text{if short/sell} \\ 1, \text{if long/buy} \end{cases}, \tag{9}$$

P_t and v_{t-i} can be calculated through the algorithm after each transaction, and at first $P_0 = 0$. In this work, the position P_t is usually calculated from a steady strategy by time step t, which means during this strategy period all the actions remain the same. In this case, the duration of a given invariant action can be calculated as $d_t = n$, and n is the same as that in Eq. (8). For all $i \le d_t$, $\omega_{t-i} = -1$, or for $i \le d_t$, $\omega_{t-i} = 1$.

4. DRL for stock price trend forecasting

The DQN is first proposed to learn successful policies for the challenging domain of classic Atari 2600 games and some challenging tasks, where three CNN layers and two full connection layers are utilized as the Q-network. However, the previous Q-Network structure in DQN is not suitable for stock price trend prediction. Therefore, a DQN variant is proposed to address this problem. Researches also indicate that DQN may suffer from overestimation for the values of the actions. Hence Double DQN is put forward to solve this issue. Inspired by Double DQN, the asynchronous m-step Double DQN is proposed for stock price trend forecasting in this section. The DQN agent is fed with the inputs presented in Section 3.2, and is set to exploit the optimized investment/trading strategy through an iterative learning algorithm. In addition, two reward functions are designed for model training. The actions, states, and rewards are described first. Then a well-designed Deep Learning Network (DLN) is introduced acting as policy work to choose actions and learn good policies. Asynchronous RL paradigm is also implemented in this work to improve the steadiness of the whole model.

4.1. Actions

The process of coming up with an investment strategy is regarded as making a sequence of actions. For the selected stock at time step t, the agent selects action i, a_t^i, from the action set $A_t = \{a_t^0, a_t^1\}$, where

$$a_t^i = \begin{cases} long/buy, & if\ i = 0 \\ short/sell, & if\ i = 1. \end{cases} \tag{10}$$

Thus, a_t^0 corresponds to long/buy the selected stock at time t; a_t^1 corresponds to short/sell the selected stock at time t. The size of action space at time step t is $|A_t| = 2$.

4.2. States

The state refers to observed state. And it is assumed that the partially observable environment in this financial market is treated as a fully observable one. Thus, the state S_t at time t consists of all the features described in Section 3.2, and is given by

$$S_t = (s_t, s_{t-1}, \ldots, s_{t-w}, P_t, d_t), \tag{11}$$

where w is the window size of the observation. And s_t is presented as

$$s_t = \nabla I_t = (\nabla Op_{t-2}, \nabla Cl_{t-2}, \nabla Hi_{t-2}, \nabla Lo_{t-2}, \nabla V_{t-2},). \tag{12}$$

At each time step t, the environment provides the current state information, based on the inputs and which action the RL agent took during the past steady strategy period.

4.3. Rewards

Reward function in RL servers as an important role as loss function in deep learning model. The design of reward function determines the learned policies and the performance of RL agent on different tasks. But the reward function in RL is more flexible than loss function in deep learning model. For example, in stock price trend prediction task, return can also be incorporated in reward function besides the prediction accuracy. Here, the different feedbacks in RL and deep learning model are compared in the following Fig. 1.

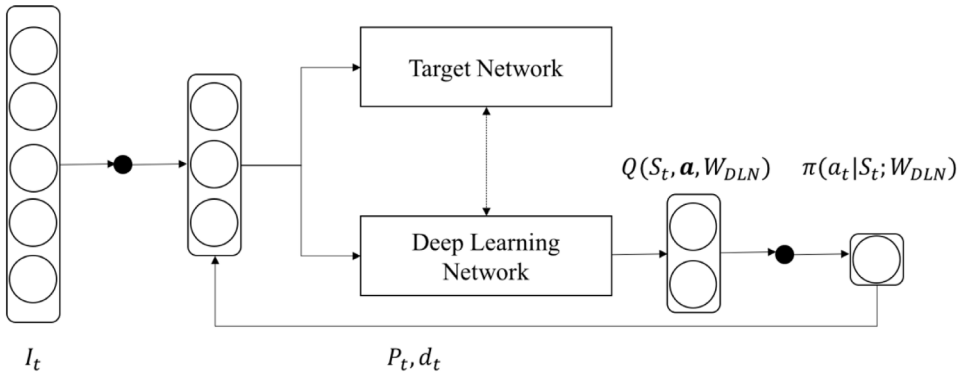

Fig. 2. The policy network architecture with Double Deep Q-Network.

In Fig. 1, the first line shows the stock price movements in the continuous three days, decrease, increase and decrease. In this environment, the stock trade is subject to T+1 settlement date with no transaction cost. Suppose there is an accuracy-oriented classifier predicts rightly and takes the trading action as short, long, short accordingly. The prediction accuracy for this classifier is 100% and this is a positive feedback to it, meaning it is a right policy for accuracy-oriented classifier. However, in stock market environment, the so-called right policy may be not a profitable policy, which means the accuracy is less important than return. As shown in the third line of Fig. 1, the reward to the action in day *t-1* is $-ln(p_t/p_{t-1})$, which is a minus; and the reward to the action in day t is $ln(p_{t+1}/p_t)$, which is also a minus. Thus, the reward to the policy "*short, long, short*" above is $ln\left(\frac{p_{t+1}p_{t-1}}{{p_t}^2}\right)$, which is a minus. This means the right policy above with 100% prediction precision is not a profitable policy. The high prediction precision does not always result in profitable policy.

Considering this situation, two different reward functions are proposed in this section. The immediate scalar reward R_t is produced as feedback of agent's action to learn a policy. It depends on whether the agent long/buy or short/sell at each execution of stock transaction. According to the definitions of action and state, the agent takes the action a_t^i at time step t depending on the state at time t. Since it is assumed the stock transactions are subject to the T+1 settlement date, the agent takes only one action at each time step. Thus, it is assumed that the execution price of the selected stock at time step t is the closing price at time t. The trading cost c, including the stamp tax, brokerage fee, exchange fee, etc., is measured in the standard financial unit of basis points. In this work, the trading cost is set to 0.5%. In a continuous steady long/buy strategy period, if $i_0 \leq n$, for a given stock holding j traded at time step $t - i_0$, the stock return at time step t is calculated as

$$r_t^j = ln\, lnCl_t - lnCl_{t-i_0} = \sum_{i=0}^{i_0-1}(\ln lnCl_{t-i} - lnCl_{t-i-1}) = \sum_{i=0}^{i_0-1}\nabla Op_{t-i} \tag{13}$$

Similarly, in a continuous steady short/sell strategy period, if $i_0 \leq n$, for a given stock holding j traded at time step $t - i_0$, the stock return at time step t is calculated as

$$r_t^j = -\left(ln\, lnCl_t - lnCl_{t-i_0}\right) = -\sum_{i=0}^{i_0-1}(\ln lnCl_{t-i} - lnCl_{t-i-1})$$
$$= -\sum_{i=0}^{i_0-1}\nabla Op_{t-i} \tag{14}$$

However, the stock return at time step t for a given stock holding j, r_t^j, cannot stand for the reward at time step t in the whole RL system. The agent may possess more than one unit of stock holding, since for a given stock, it may be traded on several different days, which means $n \geq 1$. Besides, it is free to set a reward function in a RL algorithm according to the target of the training. In this case, a well-defined reward is the pivot for RL, thus two different kinds of reward functions are devised. Similar to the discrepancy between accrual basis and cash basis in accounting principle, the difference between these two kinds of reward functions R_t at time step t is whether to take the stock return from time step t to time step $t+1$ into account.

Reward 1

As is shown in Eq. (14), the reward for action a_t^i at time step t is

$$R_t = \sum_{j=1}^{n} r_t^j - c. \tag{15}$$

Like the principle of cash basis, it is based on the real-time stock return observed on the step t. The real return of the stock holdings in position P_t (with transaction cost deducted) is regarded as the rewards from stock market environment at time step t.

Reward 2

The second kind of reward for action a_t^i at time step t is calculated as

$$R_t = \sum_{j=1}^{n} r_{t+1}^j - c. \tag{16}$$

Since it is assumed the stock transactions are subject to the T+1 settlement date, the action a_t^i at time step t will cause the stock returns susceptible to the stock price changes at the next time step $t+1$. Like the principle of accrual basis, the return receivable of the stock holdings in position P_t (with transaction cost deducted) is regarded as the rewards from stock market environment at time step t.

4.4. Policy network

After the market environment, actions, states, and rewards are introduced, the next step is to design a policy network $\pi(\boldsymbol{a}|S; W_{DLN})$ to select an action given the current state, where $\boldsymbol{a} = (a_t^0, a_t^1)$, $\boldsymbol{S}$ is the states fed to the policy network, and W_{DLN} is the weights of deep learning network in this policy network. The architecture of the policy network is shown as Fig. 2.

Previous DQN in research [51] utilizes sequential connected three CNN layers and two full connection layers as Q matrix, but

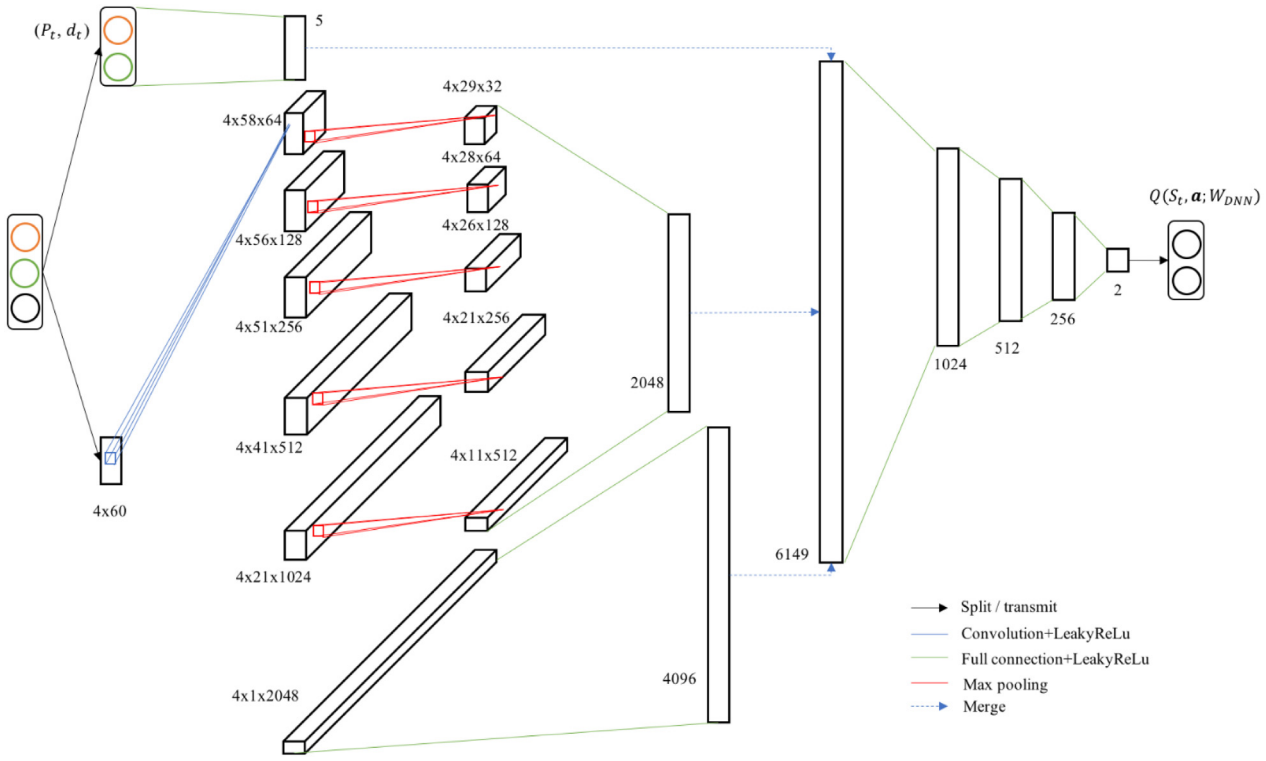

Fig. 3. The deep learning network architecture.

this architecture is not tailored for stock price trend prediction and stock trading. Thus, a DQN variant is proposed in this section.

It shows that the deep learning network is applied to estimate the policy network, called Q matrix/network. The initial input of the DQN is I_t. After the representation process, the state S_t at time step t is the real input of the deep learning network. The output of the deep learning network is $Q(S_t, \boldsymbol{a}; W_{DLN})$, the vector of action-value function, which is the expected reward starting from state S_t, taking actions $\boldsymbol{a}$, where $\boldsymbol{a} = (a_t^0, a_t^1)$. In Fig. 2, the left black point is the feature extraction process, and P_t, d_t are the selected features obtained from the algorithm outputs. The right black point is the process to choose an action under some rules. Besides, the target network and deep learning network (DLN) share the same architecture, with the same initialization. Every m steps, the target network and DLN share the same weights, which is elaborated in detail in Section 4.5. And the target return is generated from the target network. Fig. 3 illustrates the detailed architecture of deep learning network devised in our model.

As shown in Fig. 3, the input $S_t = (s_t, s_{t-1}, \ldots, s_{t-w}, P_t, d_t)$ is split into two parts, features (P_t, d_t) and features $(s_{t-1}, s_{t-2}, \ldots, s_{t-w})$. Next, six kinds of convolutional layers combined with max pooling layers are utilized to extract the relation of financial data under ith-order lag condition from features $(s_{t-1}, s_{t-2}, \ldots, s_{t-w})$. Besides, three different kinds of full connection layer are added to the features (P_t, d_t), output of Convolution 1–5 and Convolution 6 respectively; Then the outputs are merged into a high-dimensional feature vector. Then four full connection layers are added to the merged layer in proper order. At last, $Q(S_t, \boldsymbol{a}; W_{DLN})$ is the 2-dimensional output with the values representing immediate reward of two different actions. Table 1 shows the detailed parameters and architecture of the deep learning network.

As shown in Table 1, different convolutional filters with different strides are applied to extract the potential relations among the stock features from different time periods. For example, the Convolution 1 with filters (1, 3) is exploited to extract the third-order lag features between information at time step t and time step $t-3$. The architecture of the CNN part is designed under the principle of exploring relations between information with different lag orders, which is suitable for financial time series information.

4.5. Paradigm for Double Deep Q-Network with experience replay

Since the sequence of input data is non-stationary, and the online DQN updates are highly correlated [56–59], the algorithm is fundamentally unstable [55], which may cause overestimation attributing to flexible function approximation [69] and noise [70]. To this end, Double DQN [58] is introduced to decouple the action selection from the policy evaluation.

Double DQN is designed to reduce overestimations through decomposing the process of max operation in the target network into action evaluation and action selection. The target network is of the same architecture as DLN in Fig. 3 with the same initialization. Every m steps, the target network is updated with the same weights as DLN so that $W_{target} = W_{DLN}$. The output of DLN is represented as $Q(S_t, \boldsymbol{a}; W_{DLN})$ and the output of target network is $Q'\left(S_t, \boldsymbol{a}; W_{target}\right)$. Thus, according to Double DQN, the target $Y_t^{TargetDQN}$ is defined as

$$Y_t^{TargetDQN} := R_t + \gamma Q'(S_{t+1}, argmax_{a_t} Q(S_{t+1}, a_t; W_{DLN}); W_{target}). \tag{17}$$

The following is the algorithm of Double DQN with experience replay.

5. Experiments

In this section, experiments are carried out to test the performance of our proposed Double DQN. According to different experiment requirements, different stocks and stock indices are selected in different experiment parts. The experiments performed

Table 1
Parameters of deep learning network architecture.

Input S_t						
Feature (P_t, d_t)	Feature $(s_{t-1}, s_{t-2}, \ldots, s_{t-w})$					
Full Connection	Convolution 1 filters: (1,3), 64	Convolution 2 filters: (1,5), 128	Convolution 3 filters: (1,10), 256	Convolution 4 filters: (1,20), 512	Convolution 5 filters: (1,40), 1024	Convolution 6 filters: (1,60), 2048
nodes: 5 ReLu: 0.001	LeakyReLu: 0.001	LeakyReLu: 0.001	LeakyReLu: 0.001	LeakyReLu: 0.001	LeakyReLu: 0.001	LeakyReLu: 0.001
	Max Pooling: (2,2)	Max Pooling: (2,2)	Max Pooling: (2,2)	Max Pooling: (2,2)	Max Pooling: (2,2)	Max Pooling: (2,2)
			Full Connection nodes: 2048 LeakyReLu: 0.001			Full Connection nodes: 4096 LeakyReLu: 0.001
			Full Connection nodes: 1024; LeakyReLu: 0.001			
			Full Connection nodes: 512; LeakyReLu: 0.001			
			Full Connection nodes: 256; LeakyReLu: 0.001			
			Full Connection nodes: 2, LeakyReLu: 0.001			

Algorithm: Double Deep Q-Network with experience replay

Initialize replay memory D to capacity N

Initialize action-value function Q with random weights W_{DLN}

Initialize target action-value function Q' with weights W_{Target}

Initialize action set as $A = \{a^0, a^1\}$

Process sequence $\{I_t\}$ into $\{S_t\} = \{(s_t, s_{t-1}, \ldots, s_{t-w}, P_t, d_t)\}$ with window size set as w for all t

For episode = 1, M **do**

Initialize sequence $S_1 = (s_1, s_{1-1}, \ldots, s_{1-w}, P_1, d_1)$

For t = 1, T **do**

Load sequence S_t

With probability ε select a random action a_t

Otherwise, select $a_t = argmax_{a_t} Q(S_t, a_t; W_{DLN})$

Execute action a_t in emulator and observe reward R_t and processed stock sequence S_{t+1}

Update position P_t and duration d_t according to action a_t

Store transition tuple (S_t, a_t, R_t, S_{t+1}) in memory D

Sample random mini-batch of transitions $\{(S_i, a_i, R_i, S_{i+1})\}$ from memory D

Set target $Y_i^{TargetDLQN} = \begin{cases} R_i \text{ if episode terminates at step } i \\ R_i + \gamma Q'\left(S_{i+1}, argmax_{a_{i+1}} Q(S_{i+1}, a_{i+1}; W_{DLN}); W_{target}\right) \text{ otherwise} \end{cases}$

Perform a gradient descent step on $(Y_i^{TargetDQN} - Q(S_{i+1}, a_i; W_{DLN}))^2$ with respect to the network parameters W_{DLN}

Every m steps reset $Q' = Q$ through setting $W_{target} = W_{DLN}$

End For

End For

on American stock market data are within a 3-year training-period from Jan. 1st, 2015 to May 31th, 2018 and testing-period from June 1st to Sept. 24th; The ones on Chinese stock market data are within a 3-year training-period from Jan. 1st, 2015 to March 14th, 2018 and testing period from March 15th to June 15th, 2018.

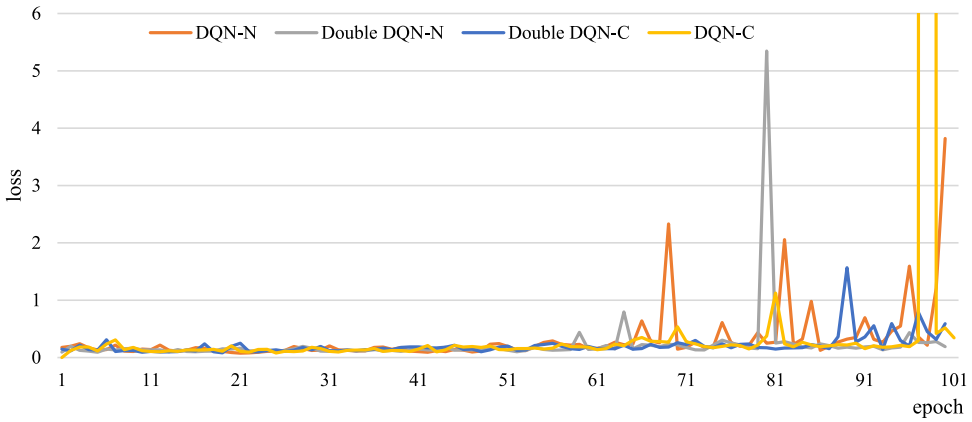

Fig. 4. Loss changes of different DRL methods on stock index GSPC.

Experiment settings

It is supposed that stock transaction is subject to T+1 settlement date. For each transaction, the trade volume executed by DRL agent is one unit. All the stocks are traded at the closing price in the period. The cost is set to 0.5% for each transaction. Besides, during the training phase, short execution is allowed and margin rate is omitted when computing the reward; while in the testing period, it changes and is illustrated in each subsection.

The input of all the Double DQN models includes closing price, volume, opening price, highest price and lowest price. A fixed 60 trading-days moving training window is chosen for each stock/index, which is followed by a one-day prediction window. The prediction is made before the market opens in every market day.

All the following experiments for the proposed model share the same parameter settings. The probability ε is set to 0.5 and gradually reduces to 0.1; the length of memory D is 700; the batch size for random sampled mini-batch transactions is 50; the discount rate γ is 0.9; the maximum number of training epochs is 100; MSE loss and RMSprop optimizer in Keras are employed with the default setting, where learning rate is 0.001 and decay is 0.9.

In addition to the policy network in Fig. 3, other different model architectures of policy network have been tested in the model selection procedure, i.e., double convolution 1–5 filters with max pooling layers, changing kernel size to $(2, i)$ for all the convolution filters (where $i \in \{3, 5, 10, 20, 40, 60\}$), or deleting full connection layer with 256 hidden nodes, etc. The experiment results suggest that the proposed policy network in this paper is an optimal model.

Baseline methods

SVM: The classic SVM model is introduced into the experiment as a baseline model, with LIBSVM designed by Lin and Chang [71] selected as the model tool. The default setting is selected; some important details are shown as: (1) setting type of SVM as C-SVC; (2) choosing radial basis function as kernel function; (3) setting gamma in kernel function as 1 divided by the number of features; (4) setting cost parameter of C as C-SVC.

LSTM: An LSTM is employed as the baseline too. The number of LSTM cells is 128; dropout rate is 0.2; the maximum number of training epochs is 50; batch size is set to 16; cross entropy loss and RMSprop optimizer is employed, where the initial learning rate is 0.001 and the decay rate is 0.99.

5.1. Results on over-optimism

In this part, four American stock indices are selected: Dow Jones Industrial Average (DJI), S&P 500 (GSPC), NASDAQ Index (IXIC), and Russell 2000 (RUT). The experiment is designed to test the overestimations of different DRL models. DQN-N stands for the DQN model with four full connection layers in Q-Network; Double DQN-N represents the Double DQN model with four full connection layers in Q-Network; DQN-C is the DQN model with Q-Network structure in Fig. 3; Double DQN-C is the Double DQN model with Q-Network structure in Fig. 3.

Table 2
Performances of Double DQN-N.

Stock index	Cash basis	
	Accuracy	Return
DJI	0.6538	0.2145
GSPC	0.6538	0.0120
IXIC	0.6154	0.0388
RUT	0.6623	−0.0183

Here, GSPC is taken as an example to show the loss changes of different DRL methods in Fig. 4 since the experiment results on four stocks are similar. As shown, the performances of DQN models based on double Q-learning algorithm are better than those with single Q-Network. Overestimation occurs, more frequent and more severe, from epoch 60 to epoch 100, where DQN-N and DQN-C are significantly unstable as the epoch increases. Double DQN-C shows the least overestimation among these four DRL methods. This result indicates Double DQN model with well-designed deep learning architecture in Fig. 3 effectively reduces the overestimation of DRL model and is more stable.

5.2. Results on a different model structure

In this part, eight stock indices are selected from American and Chinese stock markets. And the selected stock indices are Dow Jones Industrial Average (DJI), S&P 500 (GSPC), NASDAQ Index (IXIC), Russell 2000 (RUT), Shanghai Stock Exchange Index (000001), Shanghai-Shenzhen 300 (000300), Shenzhen Composite Index (399106), and CSI 500 Quality and Growth Index (000905). The following are some rules in testing period. Short execution is allowed for stock indices in American stock market; while not allowed for those in Chinese stock market. In addition, different from may researches on price trend prediction, the initial margin for short operation in NASDAQ market is also included when calculating annual return rate, and the initial margin rate for each transaction is set to 10% of the trading volume for a single transaction. Transaction cost 0.5% is considered for Double DQN, LSTM and SVM when computing annual return rate.

Table 2 shows the results of Double DQN-N on predicting the stock index. Double DQN-N does not have convolutional layer

Table 3
Performances of Double DQN-C.

Stock index	Accrual basis		Cash basis		B&H
	Accuracy	Return	Accuracy	Return	
DJI	0.5513	2.3433	0.6282	1.3475	0.2934
GSPC	0.6154	0.5582	0.6538	0.0833	0.2445
IXIC	0.6026	0.1905	0.6282	0.0944	0.1964
RUT	0.5844	0.0771	0.6623	0.0074	0.1339
000001	0.5410	0.0657	0.5902	0.0271	−0.3272
000300	0.5806	0.7402	0.6129	0.0606	−0.3347
000905	0.6613	10.5220	0.6452	0.3714	−0.4506
399106	0.6129	9.8969	0.6290	0.4473	−0.3900

Table 4
Performances of LSTM and SVM.

Stock index	LSTM		SVM		B&H
	Accuracy	Return	Accuracy	Return	
DJI	0.6049	0.1241	0.59259	−0.0413	0.2934
GSPC	0.5679	−0.0455	0.5679	−0.0420	0.2445
IXIC	0.5926	0.1608	0.59259	−0.0399	0.1964
RUT	0.5679	0.0147	0.5679	−0.0360	0.1339
000001	0.5938	0	0.41935	0.2609	−0.3272
000300	0.5625	0	0.43548	−0.0507	−0.3347
000905	0.5938	0	0.41935	−0.0633	−0.4506
399106	0.5625	0	0.45161	−0.0617	−0.3900

Table 5
Stock information.

NASDAQ stock	Max drawdown	SSE stock	Max drawdown
1	0.0480	1	0.1936
2	0.0626	2	0.0943
3	0.1636	3	0.2328
4	0.2345	4	0.2113
5	0.1107	5	0.0860
6	0.0253	6	0.1487
7	0.0718	7	0.4554
8	0.0493	8	0.1355
9	0.1916	9	0.3036
10	0.3065	10	0.2435

and max pooling layer in comparison with Double DQN-C. Cash basis reward function is used in this model, which is the same as Eq. (15). Accuracy and annual return are utilized to evaluate the performances of Double DQN-N.

Table 3 shows the results on forecasting both American and Chinese stock indices based on different reward functions, namely accrual basis (Eq. (16)) and cash basis (Eq. (15)). B&H refers to the buy and hold strategy. Similarly, we use accuracy and return to measure the performances of Double DQN-C. In general, accrual basis reward function has a better performance than cash basis reward function in terms of return, whereas its accuracy is relatively lower in most cases compared with cash basis reward function. These results coincide with our theoretical expectations when designing the two different reward functions. It also demonstrates that reward function is of great importance in determining the performances of Double DQN model. Besides, both of the Double DQN-C models in Table 3 learn a better trading strategy in contrast to the B&H strategy in most cases. As shown in Tables 2 and 3, Double DQN-C model gets a higher return than DQN-N model without reducing the accuracy. This indicates that convolutional layer and max pooling layer structure designed in this work captures the hidden dependencies and latent dynamics in the stock index data, which does benefit the prediction performance.

In Table 4, baseline methods LSTM and SVM are used to predict both American and Chinese stock indices. LSTM model does well in forecasting the fluctuation of stock indices. However, its performances are poor in terms of return rate. As for SVM, its accuracy on predicting the American stock indices is high. Nevertheless, it is not good at forecasting Chinese stock indices. Similarly, the annual returns of SVM are not as high as those of Double DQN model for almost all the stock indices. Moreover, both baselines cannot compete with B&H in terms of return when forecasting American stock indices. As shown in Tables 3 and 4, compared to baselines, Double DQN models show superiority on both indicators.

5.3. Quality of learned policies on stocks

Massive empirical analysis experiments on both NASDAQ[2] and Shanghai Stock Exchange (SSE)[3] are implemented to test the effectiveness and practicability of the proposed Double DQN for stock market prediction. In this subsection, Double DQN refers to Double DQN-C since Double DQN-N is not discussed any more. The results reported below are obtained by applying the aforementioned model to ten daily-based NASDAQ stocks and ten daily-based SSE stocks. All the stocks are randomly selected. The information of selected stocks is shown as follows:

Table 5 shows that the max drawdown ranges from 0.0480 to 0.3065 for NASDAQ stocks and ranges from 0.0860 to 0.4554 for SEE stocks. This means stocks with different risks are selected to test the performance of the proposed model.

Different from the training phase, some rules change in the testing period. For stocks in Shanghai Stock Exchange, short execution is not allowed; these data are trained through Double DQN with accrual basis reward function. For stocks in NASDAQ, short execution is allowed; and the data are trained via Double DQN with cash basis reward function. The initial margin rate for each transaction is set to 20% of the trading volume for a single transaction. Transaction cost 0.5% is considered for both Double DQN and SVM when computing annual return rate.

As shown in Table 6, for the selected stocks from NASDAQ, the highest accuracy of Double DQN is 65.79%, and the highest revenue for a stock obtained by the model is 10.7137 units. In order to compare the performance of different models, the annual return rate is calculated and shown as follows. The maximum

[2] https://www.nasdaq.com/.
[3] http://www.sse.com.cn/.

Table 6
Performances of Double DQN on NASDAQ stocks.

Stock ID	Accuracy	Revenue	Return	Stock ID	Accuracy	Revenue	Return
1	0.6265	1.1784	0.1992	6	0.6329	1.9402	0.3535
2	0.6410	4.3086	0.4501	7	0.6184	2.2957	0.3922
3	0.6076	0.7599	0.1176	8	0.6282	10.7137	2.1692
4	0.6579	1.0353	0.2878	9	0.5641	0.3731	0.0827
5	0.6329	3.8013	0.5469	10	0.6145	0.0631	0.0033

Table 7
Performances of SVM on NASDAQ stocks.

Stock ID	Accuracy	Revenue	Return	Stock ID	Accuracy	Revenue	Return
1	0.5395	7.7471	1.6717	6	0.4342	1.3432	0.2977
2	0.5867	2.4487	0.5149	7	0.4933	2.7106	0.5927
3	0.5395	0.7098	0.1532	8	0.5867	5.4579	1.1935
4	0.5132	5.9476	0.2567	9	0.4800	5.2263	0.2256
5	0.4211	0.3323	0.0717	10	0.5132	−0.4568	−0.1441

Table 8
Performances of Double DQN on SSE stocks.

Stock ID	Accuracy	Revenue	Return	Stock ID	Accuracy	Revenue	Return
1	0.7018	2.0985	0.6995	6	0.6939	0.1003	0.0573
2	0.6346	2.9997	1.0908	7	0.6071	0.3877	0.0470
3	0.6610	0.5616	0.0775	8	0.6667	0.8531	0.2275
4	0.6842	0.0146	0.0292	9	0.6667	0	0
5	0.6271	0.9151	0.2034	10	0.661	0.0679	0.0543

Table 9
Performances of SVM on SSE stocks.

Stock ID	Accuracy	Revenue	Return	Stock ID	Accuracy	Revenue	Return
1	0.4561	0	0	6	0.4364	−3.2562	−0.2368
2	0.4909	1.4996	0.1091	7	0.4727	−2.7035	−0.4506
3	0.5000	−1.5534	−0.1071	8	0.4483	0.4284	0.0295
4	0.5789	0	0	9	0.3269	−7.6029	−0.5848
5	0.4746	3.9846	0.2701	10	0.4746	−0.5613	−0.7484

return rate is 216.92% for the selected stocks from NASDAQ. However, the highest accuracy of SVM is 58.67%, the highest revenue is 7.7471 units and the maximum annual return rate is 167.17%.

It shows that Double DQN outperforms SVM in terms of accuracy on NASDAQ stocks. Double DQN shows a better performance in general with respect to the return. Although SVM gains better return rate for stocks 1, 2, 3, 7, 9, Double DQN is better in the other five stocks; besides, the SVM made a failure strategy for stock 10, which gets negative return rate.

Table 8 indicates that the highest accuracy of Double DQN is 70.18%, and the highest revenue for a stock obtained by the model is 2.9997 units. The maximum return rate is 109.08% for the selected stocks from SSE. However, the highest accuracy of SVM is 57.89%; the highest revenue is 3.9846 units; the maximum annual return rate is 27.01%. It shows that Double DQN outperforms SVM in terms of accuracy on SSE stocks. Double DQN has a better performance in general with respect to the return. Although SVM gains higher revenue on stock 5, Double DQN does better on the other nine stocks. Besides, SVM makes a failure strategy on stocks 3, 6, 7, 9, 10, with negative return rates. In addition, the revenue obtained by SVM strategy on stock 5 amounts to 3.9846 units but this mainly benefits from continuous "buy" operation, which requires larger amounts of capital. Thus, the return rate of SVM on stock 5 is only 27.01%.

The results in Tables 6 to 9 show that high accuracy and high return rate obtained by a given model are usually not concurrent. This partly depends on the general trend of the selected stocks during the testing period. For example, the max drawdown of stock 10 in NASDAQ is 30.65%, which means it is a high-risk stock. In this case, if a model can successfully avoid loss, i.e. gaining negative return rate or negative revenue, it is considered a good agent. This reveals that although the DRL agent does not gain good return rate, it still gets a good performance avoiding loss if a risky stock, such as stock 10 in NASDAQ or stock 9 in SSE, is selected. Furthermore, since the task of Double DQN is to maximize the expected reward of each action based on the current state, different from the objective of the traditional machine learning method, the high revenue does not necessarily mean high accuracy. Subject to the objective function defined in this work, the DRL agent tends to take right action (which means to take risks) when the price fluctuates dramatically instead of pursuing a higher total hit rate.

6. Conclusion

A Double DQN based paradigm is proposed for stock price trend prediction and stock trading in this work. It shows that the deep learning architecture with CNN layers in policy network is capable of exploring and extracting the latent dependency in the stock data. The experiment also indicates the proposed Double DQN reduces overestimation and improves the stability of the model. It also shows that the proposed method does well in pursuing high revenue via transaction strategy generated from DRL agent. In addition, the Double DQN trained on accrual basis reward function gets a higher return and that on cash basis reward function performs better in accuracy. It also reveals the difference between the proposed method and traditional machine learning methods through the comparison of the objective function, explaining why Double DQN could gain high revenue with relatively low hit rate. This gives instructions for the investor to make more profitable and reasonable investment strategy.

Also, there are still some limitations in this work. It can be extended to the application of other reinforcement algorithms;

More stock data are supposed to be tested in the experiment; And it is advisable to compare the performances of models with different reward functions.

CRediT authorship contribution statement

Yong Shi: Resources, Supervision, Funding acquisition, Formal analysis. **Wei Li:** Conceptualization, Investigation, Validation, Writing - review & editing. **Luyao Zhu:** Methodology, Software, Investigation, Writing - original draft, Formal analysis. **Kun Guo:** Data curation, Visualization, Project administration. **Erik Cambria:** Writing - review & editing, Resources, Funding acquisition.

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.

Acknowledgments

This research is supported by the National Natural Science Foundation of China No. 71932008, and the Agency for Science, Technology and Research (A*STAR), Singapore under its AME Programmatic Funding Scheme (Project #A18A2b0046).

References

[1] S. Thawornwong, D. Enke, The adaptive selection of financial and economic variables for use with artificial neural networks, Neurocomputing 56 (2004) 205–232, http://dx.doi.org/10.1016/j.neucom.2003.05.001.
[2] Q. Cao, K.B. Leggio, M.J. Schniederjans, A comparison between Fama and French's model and artificial neural networks in predicting the Chinese stock market, Comput. Oper. Res. 32 (2005) 2499–2512.
[3] D. Enke, N. Mehdiyev, Stock market prediction using a combination of stepwise regression analysis, differential evolution-based fuzzy clustering, and a fuzzy inference neural network, Intell. Autom. Soft Comput. 19 (2013) 636–648, http://dx.doi.org/10.1080/10798587.2013.839287.
[4] A.M. Rather, A. Agarwal, V.N. Sastry, Recurrent neural network and a hybrid model for prediction of stock returns, Expert Syst. Appl. 42 (2015) 3234–3241, http://dx.doi.org/10.1016/j.eswa.2014.12.003.
[5] E. Chong, C. Han, F.C. Park, Deep learning networks for stock market analysis and prediction, Expert Syst. Appl. 83 (2017) 187–205.
[6] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks\r, Science (80-.) 313 (2006) 504–507, http://dx.doi.org/10.1126/science.1127647.
[7] H. Lee, P. Pham, Y. Largman, A. Ng, Unsupervised feature learning for audio classification using convolutional deep belief networks, Neural Inf. Process. Syst. (2009) 1–9, http://dx.doi.org/10.1145/1553374.1553453.
[8] D. Cireşan, U. Meier, J. Masci, J. Schmidhuber, Multi-column deep neural network for traffic sign classification, Neural Netw. 32 (2012) 333–338, http://dx.doi.org/10.1016/j.neunet.2012.02.023.
[9] G.S. Atsalakis, K.P. Valavanis, Stock market forecasting part I: Conventional models, (2013) 49–104.
[10] A. Charles, O. Darné, Large shocks and the September 11th terrorist attacks on international stock markets, Econ. Model. 23 (2006) 683–698, http://dx.doi.org/10.1016/j.econmod.2006.03.008.
[11] Y. Hu, B. Feng, X. Zhang, E.W.T. Ngai, M. Liu, Stock trading rule discovery with an evolutionary trend following model, Expert Syst. Appl. 42 (2014) 212–222, http://dx.doi.org/10.1016/j.eswa.2014.07.059.
[12] A. Assaf, Dependence and mean reversion in stock prices: The case of the MENA region, Res. Int. Bus. Financ. 20 (2006) 286–304, http://dx.doi.org/10.1016/j.ribaf.2005.05.004.
[13] G. Dufrénot, D. Guégan, A. Péguin-Feissolle, Long-memory dynamics in a SETAR model - Applications to stock markets, J. Int. Financ. Mark. Institutions Money 15 (2005) 391–406, http://dx.doi.org/10.1016/j.intfin.2004.09.001.
[14] I. Virtanen, P. Yli-Olli, Forecasting stock market prices in a thin security market, Omega 15 (1987) 145–155, http://dx.doi.org/10.1016/0305-0483(87)90029-6.
[15] M. Kavussanos, I. Visvikis, The predictability of non-overlapping forecasts: Evidence from a new market, Multinatl. Financ. J. 15 (2011) 125–156.
[16] E. Balaban, A. Bayar, R.W. Faff, Forecasting stock market volatility: Further international evidence, Eur. J. Financ. 12 (2006) http://dx.doi.org/10.1080/13518470500146082.
[17] W. Huang, Y. Nakamori, S.Y. Wang, Forecasting stock market movement direction with support vector machine, Comput. Oper. Res. 32 (2005) 2513–2522, http://dx.doi.org/10.1016/j.cor.2004.03.016.
[18] D.W. Jansen, Z. Wang, Evaluating the fed model of stock price valuation: An out-of-sample forecasting perspective, Adv. Econom. 20 (PART 2) (2006) 179–204, http://dx.doi.org/10.1016/S0731-9053(05)20026-9.
[19] M.T. Bradshaw, L.D. Brown, K. Huang, Do sell-side analysts exhibit differential target price forecasting ability? Rev. Account. Stud. 18 (2013) 930–955, http://dx.doi.org/10.1007/s11142-012-9216-5.
[20] H.M. Krolzig, J. Toro, Multiperiod forecasting in stock markets: A paradox solved, Decis. Support Syst. 37 (2004) 531–542, http://dx.doi.org/10.1016/S0167-9236(03)00085-X.
[21] K. Zhang, L. Chan, Efficient factor GARCH models and factor-DCC models, Quant. Finance (2009) http://dx.doi.org/10.1080/14697680802039840.
[22] P. Giot, S. Laurent, Modelling daily Value-at-Risk using realized volatility and ARCH type models, J. Empir. Financ. 11 (2004) 379–398, http://dx.doi.org/10.1016/j.jempfin.2003.04.003.
[23] J. Tolvi, Long memory in a small stock market, Econ. Bull. 7 (2003).
[24] A. Black, P. Fraser, N. Groenewold, U.S. stock prices and macroeconomic fundamentals, Int. Rev. Econ. Financ. 12 (2003) 345–367, http://dx.doi.org/10.1016/S1059-0560(03)00016-9.
[25] P.G. Patev, N.K. Kanaryan, Modelling and Forecasting the Volatility of Thin Emerging Stock Markets: The Case of Bulgaria, 2004, http://dx.doi.org/10.2139/ssrn.532302.
[26] H. White, Economic prediction using neural networks: The case of IBM daily stock returns, in: IEEE Int. Conf., Neural Netw. (1988) 451–458, http://dx.doi.org/10.1109/ICNN.1988.23959.
[27] W.C. Chiang, D. Enke, T. Wu, R. Wang, An adaptive stock index trading decision support system, Expert Syst. Appl. 59 (2016) 195–207, http://dx.doi.org/10.1016/j.eswa.2016.04.025.
[28] X. Zhong, D. Enke, Forecasting daily stock market return using dimensionality reduction, Expert Syst. Appl. 67 (2017) 126–139, http://dx.doi.org/10.1016/j.eswa.2016.09.027.
[29] E. Chong, C. Han, F.C. Park, Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies, Expert Syst. Appl. 83 (2017) 187–205, http://dx.doi.org/10.1016/j.eswa.2017.04.030.
[30] S. Selvin, R. Vinayakumar, E.A. Gopalakrishnan, V.K. Menon, K.P. Soman, Stock price prediction using LSTM, RNN and CNN-sliding window model, in: 2017 Int. Conf. Adv. Comput. Commun. Informatics, ICACCI 2017, 2017, pp. 1643–1647, http://dx.doi.org/10.1109/ICACCI.2017.8126078.
[31] M. Ballings, D. Van Den Poel, N. Hespeels, R. Gryp, Evaluating multiple classifiers for stock price direction prediction, Expert Syst. Appl. 42 (2015) 7046–7056, http://dx.doi.org/10.1016/j.eswa.2015.05.013.
[32] 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, http://dx.doi.org/10.1016/j.eswa.2016.01.018.
[33] K. Zbikowski, Using volume weighted support vector machines with walk forward testing and feature selection for the purpose of creating stock trading strategy, Expert Syst. Appl. 42 (2015) 1797–1805, http://dx.doi.org/10.1016/j.eswa.2014.10.001.
[34] D. Kumar, S.S. Meghwani, M. Thakur, Proximal support vector machine based hybrid prediction models for trend forecasting in financial markets, J. Comput. Sci. 17 (2016) 1–13, http://dx.doi.org/10.1016/j.jocs.2016.07.006.
[35] H. Qu, Y. Zhang, A new kernel of support vector regression for forecasting high-frequency stock returns, Math. Probl. Eng. 2016 (2016) http://dx.doi.org/10.1155/2016/4907654.
[36] J. Patel, S. Shah, P. Thakkar, K. Kotecha, Predicting stock market index using fusion of machine learning techniques, Expert Syst. Appl. 42 (2015) 2162–2172, http://dx.doi.org/10.1016/j.eswa.2014.10.031.
[37] S. Choudhury, S. Ghosh, A. Bhattacharya, K.J. Fernandes, M.K. Tiwari, A real time clustering and SVM based price-volatility prediction for optimal trading strategy, Neurocomputing 131 (2014) 419–426, http://dx.doi.org/10.1016/j.neucom.2013.10.002.
[38] B.M. Henrique, V.A. Sobreiro, H. Kimura, Stock price prediction using support vector regression on daily and up to the minute prices, J. Financ. Data Sci. 4 (2018) 183–201, http://dx.doi.org/10.1016/j.jfds.2018.04.003.
[39] F.Z. Xing, E. Cambria, R.E. Welsch, Natural language based financial forecasting: a survey, Artif. Intell. Rev. (2018) http://dx.doi.org/10.1007/s10462-017-9588-9.
[40] X. Ding, Y. Zhang, T. Liu, J. Duan, Deep learning for event-driven stock prediction, IJCAI-15 Int. Jt. Conf. Artif. Intell. (2015) 2327–2333.
[41] F.Z. Xing, E. Cambria, R.E. Welsch, Intelligent asset allocation via market sentiment views, IEEE Comput. Intell. 13 (2018) 1–20, http://dx.doi.org/10.1109/MCI.2018.2866727.
[42] Y. Ma, H. Peng, E. Cambria, Targeted Aspect-Based Sentiment Analysis via Embedding Commonsense Knowledge into an Attentive LSTM, Aaai-2018, 2018, pp. 5876–5883, http://sentic.net/sentic-lstm.pdf.

[43] A. Picasso, S. Merello, Y. Ma, L. Oneto, E. Cambria, Technical analysis and sentiment embeddings for market trend prediction, Expert Syst. Appl. (2019) http://dx.doi.org/10.1016/j.eswa.2019.06.014.
[44] J. Hu, M.P. Wellman, Multiagent reinforcement learning: Theoretical framework and an algorithm, in: Proc. Fifteenth Int. Conf. Mach. Learn., 1998, p. 250.
[45] L.P. Kaelbling, M.L. Littman, A.W. Moore, Reinforcement learning: A survey, J. Artificial Intelligence Res. 4 (1996) 237–285, http://dx.doi.org/10.1613/jair.301.
[46] R.S. Sutton, A.G. Barto, Reinforcement learning: An introduction, IEEE Trans. Neural Netw. 9 (1998) 1054, http://dx.doi.org/10.1109/TNN.1998.712192.
[47] M. Tan, Multi-agent reinforcement learning: Independent vs. Cooperative agents, in: Mach. Learn. Proc. 1993, 1993, pp. 330–337, http://dx.doi.org/10.1016/B978-1-55860-307-3.50049-6.
[48] C.J.C.H. Watkins, P. Dayan, Q-learning, Mach. Learn. 8 (1992) 279–292, http://dx.doi.org/10.1007/BF00992698.
[49] C.J.C.H. Watkins, P. Dayan, Technical note: Q-learning, Mach. Learn. 8 (1992) 279–292, http://dx.doi.org/10.1023/A:1022676722315.
[50] V.R. Konda, J.N. Tsitsiklis, Actor-critic algorithms, Control Optim. 42 (2003) 1143–1166, http://dx.doi.org/10.1137/S0363012901385691.
[51] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, Human-level control through deep reinforcement learning, Nature 518 (2015) 529–533, http://dx.doi.org/10.1038/nature14236.
[52] J.L. McClelland, B.L. McNaughton, R.C. O'Reilly, Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory, Psychol. Rev. 102 (1995) 419–457, http://dx.doi.org/10.1037/0033-295X.102.3.419.
[53] J. O'Neill, B. Pleydell-Bouverie, D. Dupret, J. Csicsvari, Play it again: reactivation of waking experience and memory, Trends Neurosci. 33 (2010) 220–229, http://dx.doi.org/10.1016/j.tins.2010.01.006.
[54] L. Lin, Reinforcement Learning for Robots using Neural Networks, Report, C, 1993, pp. 1–155.
[55] V. Mnih, A.P. Badia, M. Mirza, A. Graves, T.P. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu, Asynchronous methods for deep reinforcement learning, 48 (2016). http://dx.doi.org/10.1177/0956797613514093.
[56] M. Riedmiller, Neural fitted Q iteration - First experiences with a data efficient neural Reinforcement Learning method, in: Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), 2005, pp. 317–328, http://dx.doi.org/10.1007/11564096_32.
[57] V. Mnih, D. Silver, M. Riedmiller, Playing atari with deep reinforcement learning (DQN), Neural Inf. Process. Syst. (2013) 1–9, http://dx.doi.org/10.1038/nature14236.
[58] H. van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double Q-learning, (2015) 2094–2100, http://dx.doi.org/10.1016/j.artint.2015.09.002.
[59] J. Schulman, S. Levine, P. Moritz, M. Jordan, P. Abeel, Trust region policy optimisation, in: ICML, 2015, http://dx.doi.org/10.1063/1.4927398.
[60] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, 2015, pp. 1–14, http://dx.doi.org/10.1561/2200000006, arXiv prepr. arXiv:1509.02971.
[61] M.G. Bellemare, Y. Naddaf, J. Veness, M. Bowling, The arcade learning environment: An evaluation platform for general agents, in: IJCAI Int. Jt. Conf. Artif. Intell., 2015, pp. 4148–4152, http://dx.doi.org/10.1613/jair.3912.
[62] H. Guo, Generating text with deep reinforcement learning, (2015) 1–10.
[63] K. Narasimhan, T. Kulkarni, R. Barzilay, Language understanding for text-based games using deep reinforcement learning, in: Emnlp2015, 2015, p. 10, http://dx.doi.org/10.18653/v1/D15-1001.
[64] J. Lee, Stock price prediction using reinforcement learning, in: Ind. Electron. Proceedings. ISIE 2001. IEEE Int. Symp., vol. 1, 2001, pp. 690–695, http://dx.doi.org/10.1109/ISIE.2001.931880.
[65] A.V. Rutkauskas, T. Ramanauskas, Building an artificial stock market populated by reinforcement-learning agents, J. Bus. Econ. Manag. 10 (2009) 329–341, http://dx.doi.org/10.3846/1611-1699.2009.10.329-341.
[66] Y. Nevmyvaka, Y. Feng, M. Kearns, Reinforcement learning for optimized trade execution, in: Proc. 23rd Int. Conf. Mach. Learn. - ICML '06, 2006, pp. 673–680, http://dx.doi.org/10.1145/1143844.1143929.
[67] L. Hailin, C.H. Dagli, D. Enke, Short-term stock market timing prediction under reinforcement learning schemes, in: Proc. 2007 IEEE Symp. Approx. Dyn. Program. Reinf. Learn. ADPRL 2007, 2007, pp. 233–240, http://dx.doi.org/10.1109/ADPRL.2007.368193.
[68] Z. Tan, C. Quek, P.Y.K. Cheng, Stock trading with cycles: A financial application of ANFIS and reinforcement learning, Expert Syst. Appl. 38 (2011) 4741–4755, http://dx.doi.org/10.1016/j.eswa.2010.09.001.
[69] S. Thrun, A. Schwartz, Issues in using function approximation for reinforcement learning, in: Proc. 4th Connect. Model. Summer Sch. Hillsdale, Lawrence Erlbaum, NJ, 1993, pp. 1–9.
[70] H. Van Hasselt, A.C. Group, C. Wiskunde, Double Q-learning, Neural Inf. Process. Syst. (2010) 1–9.
[71] C. Chang, C. Lin, LIBSVM: A library for support vector machines, ACM Trans. Intell. Syst. Technol. 2 (2011) 1–39, http://dx.doi.org/10.1145/1961189.1961199.