Information Sciences 538 (2020) 142–158

Contents lists available at ScienceDirect

Information Sciences

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

Adaptive stock trading strategies with deep reinforcement learning methods

Xing Wu [a,b], Haolei Chen [a], Jianjia Wang [a,b], Luigi Troiano [c], Vincenzo Loia [d], Hamido Fujita [e,f,g,*,1]

[a] School of Computer Engineering and Science, Shanghai University, Shanghai, China
[b] Shanghai Institute for Advanced Communication and Data Science, Shanghai University, Shanghai, China
[c] Department of Engineering, University of Sannio, Benevento 82100, Italy
[d] Department of Innovation Systems, University of Salerno, Fisciano84084, Italy
[e] Faculty of Information Technology, Ho Chi Minh City University of Technology (HUTECH), Ho Chi Minh City, Viet Nam
[f] Andalusian Research Institute on Data Science and Computational Intelligence (DaSCI), University of Granada, Spain
[g] Faculty of Software and Information Science, Iwate Prefectural University (IPU), Iwate, Japan

ARTICLE INFO

Article history:
Received 4 January 2020
Received in revised form 5 May 2020
Accepted 13 May 2020
Available online 13 June 2020

Keywords:
Stock trading strategy
Gated recurrent unit
Deep Q-learning
Deep deterministic policy gradient

ABSTRACT

The increasing complexity and dynamical property in stock markets are key challenges of the financial industry, in which inflexible trading strategies designed by experienced financial practitioners fail to achieve satisfactory performance in all market conditions. To meet this challenge, adaptive stock trading strategies with deep reinforcement learning methods are proposed. For the time-series nature of stock market data, the Gated Recurrent Unit (GRU) is applied to extract informative financial features, which can represent the intrinsic characteristics of the stock market for adaptive trading decisions. Furthermore, with the tailored design of state and action spaces, two trading strategies with reinforcement learning methods are proposed as GDQN (Gated Deep Q-learning trading strategy) and GDPG (Gated Deterministic Policy Gradient trading strategy). To verify the robustness and effectiveness of GDQN and GDPG, they are tested both in the trending and in the volatile stock market from different countries. Experimental results show that the proposed GDQN and GDPG not only outperform the Turtle trading strategy but also achieve more stable returns than a state-of-the-art direct reinforcement learning method, DRL trading strategy, in the volatile stock market. As far as the GDQN and the GDPG are compared, experimental results demonstrate that the GDPG with an actor-critic framework is more stable than the GDQN with a critic-only framework in the ever-evolving stock market.

1. Introduction

The financial industry exists to make profits by allocating resources to where they are most effective. The stock market has been a very prominent subject for individuals who want to secure investments with an opportunity to realize profit maximization. Thus, investors used to trade stocks based on their perceptions of the stock market. However, this way of trading is usually inefficient and prone to major losses caused by investors' irrational behaviors. With the rapid development of telecommunication and computer technology, tremendous revolutions are taking place in conventional stock trading, which

* Corresponding author at: Faculty of Information Technology, Ho Chi Minh City University of Technology (HUTECH), Ho Chi Minh City, Viet Nam.
E-mail addresses: h.fujita@hutech.edu.vn, HFujita-799@acm.org (H. Fujita).
[1] ORCID(s): 0000-0001-5256-210X.

https://doi.org/10.1016/j.ins.2020.05.066
0020-0255/

has encouraged highly productive developments in quantitative trading [10]. Different from typical trading behaviors, quantitative trading is a mode of profit mining from historical market data, which can be realized automatically by algorithms. Thus we are motivated by the call to develop quantitative trading strategies adapting to the ever-evolving stock market. The investment to stock markets aims to realize profit maximization, whereas the objective of reinforcement learning methods is to maximize some portion of the cumulative reward. It is promising to propose adaptive stock trading strategies with deep reinforcement learning methods.

The quantitative trading is a process to generate trading signals in a complex market environment with risk management. This process is well depicted as an online decision making according to the fluctuation of stocks. In accordance with the requirements of quantitative trading, reinforcement learning methods work as a direct adaptive optimal control of nonlinear systems. Reinforcement learning methods can be divided into three types: (1) critic-only [39]; (2) actor-only [30]; and (3) actor-critic methods [15]. The critic-only methods such as TD-learning [35] and Q-learning [21] are utilized to solve the problems of optimization in discrete space. The actor-only methods, for example, policy gradient [16], directly learn the parameterized policy with a spectrum of continuous actions. The actor-critic methods combine two aforementioned critic-only and actor-only frameworks altogether, in which the actor computes a parameterized policy and the critic evaluates such actions for the purpose of maximizing the ultimate rewards.

The agent in reinforcement learning is the component that makes the decision on what actions to take to collect as much reward as possible. That is to say, the fundamental principle of reinforcement learning is to maximize the agent's cumulative reward value from the learning process, which drives the agent to learn the optimal strategy for various types of problems. Reinforcement learning has been successfully applied in different tasks, such as unmanned driving [31], job scheduling [5], and game playing [23]. In some cases, reinforcement learning has even surpassed human experts, such as AlphaZero [24]. Therefore, reinforcement learning methods are natural choices for quantitative stock trading strategies.

For stock markets, market data is complex, noisy, nonlinear, and non-stationary. When facing a complicated market environment, agents usually cannot achieve an optimal strategy. To address this problem, deep neural networks have been employed in our proposed methods. Previous researches demonstrated that deep neural networks can be effectively integrated into reinforcement learning methods [17,13]. This is because deep neural networks [33] can extract abstract and sophisticated features from different categories of raw data. Deep learning has achieved remarkable success in many fields, such as image processing [9,40], speech recognition [1] and natural language processing [12,37]. At the same time, deep learning has been proven to be a powerful tool for effectively extracting market characteristics [2]. Therefore, the deep learning methods are utilized to extract useful representative features from stock data depicting the fluctuations of stock markets.

The main contributions of our work can be summarized as follows:

1. It is the GRU (Gated Recurrent Unit) that is proposed to extract informative features from raw financial data and technical indicators to improve the accuracy and robustness for the representation of stock market conditions.
2. A novel reward function is designed with risk-adjusted ratio for the proposed GDQN and GDPG trading strategies to ensure stable returns even in the volatile stock market.
3. With the tailored design of state and action spaces, two adaptive stock trading strategies GDQN and GDPG are proposed for quantitative stock trading, which outperform the Turtle trading strategy and achieve more stable returns than the DRL trading strategy, a state-of-the-art trading strategy with reinforcement learning method.

2. Related works

Recent developments in reinforcement learning and deep learning have brought new ideas into stock markets. According to different trading strategies, there are two streams of research about stock trading with reinforcement learning or deep learning methods. On the one hand, deep learning methods are utilized to predict the return or movement of stock markets. For example, Qiu et al. utilized a combination of neural networks and dimension-reduction techniques to forecast the daily direction of stock markets [22]. Zhong et al. applied an artificial neural network to predict the return of the Japanese stock market [41]. Some deep learning methods were also used to predict stock prices [11,38]. The predicting outcomes were incorporated into quantitative trading frameworks to improve the effectiveness of proposed trading strategy. On the other hand, some researchers were committed to building automated trading strategies. Moody et al. built a stock trading system with reinforcement learning [19], in which the actual input of the neural network is raw financial data. Neves et al. described a system for short-term speculation in the foreign exchange market, based on reinforcement learning developments [3]. Sun et al. focused on how to effectively construct dynamic financial distress prediction models for improving corporate financial risk management [26,25]. Yue et al. proposed a sparse coding-inspired optimal trading system for real-time high-frequency financial signal representation and trading [7]. They combined sparse coding and reinforcement learning to build trading strategies, where sparse coding was used for feature learning and the reinforcement learning model was the actor-only method. Troiano et al. trained a robot with long short-term memory machine which can replicate the logic underlying a strategy [27]. All in all, deep learning architectures have been proven to be effective and efficient to predict the price of stocks or make trading decisions.

However, previous studies did not fully exploit the properties of financial data and had unstable returns in a volatile market. The financial data contains a large amount of noise, jump, and movement leading to the highly nonstationary condition. To build adaptive stock trading strategies for mitigating data noise and uncertainty, we utilize technical indicators to repre-

sent stock market conditions. Due to the time-series nature of financial data, a deep learning method is applied to extract high-level features of stock markets. To be specific, the recurrent neural network (RNN) [4] and convolutional neural network (CNN) [34] are employed. Long short-term memory (LSTM) [42] is a special kind of recurrent netural network and it has been proven to be effective in time series modeling and prediction [20,8]. A more recent alternative to LSTM is the GRU (gated recurrent unit) [32]. Thus we propose to utilize GRU to extract informative features of stock data.

Following feature extraction with GRU, the reinforcement learning module will interact with the abovementioned deep learning module and make trading decisions. To be specific, the trading strategies are built with two different reinforcement learning frameworks. The proposed GDQN is a critic-only method with deep Q-network [28]. The proposed GDPG is an actor-critic method with deep deterministic policy gradient [14]. To sum up, GRU is proposed to summarize the market conditions from raw data and technical indicators of stock markets. Furthermore, critic-only GDQN and actor-critic GDPG are proposed to make trading decisions accordingly.

3. Problem formulation

The essence of reinforcement learning is that agents constantly interact with the environment and learn the optimal strategy to address specific problems. For stock trading, the environment consists of current stock market movements and historical price series, including a wide variety of raw market data as well as technical indicators. Thus, choosing an appropriate set of raw data and technical indicators is a prerequisite for agents to observe the stock market environment and learn strategies for stock trading. To source agents with informative features, the GRU is employed to model correlations between data and the stock market conditions. Afterwards, reinforcement learning methods make trading actions on the features learned by the GRU.

3.1. Depiction of stock markets

The core challenge of the stock trading is to capture the right trading time according to the market conditions and make the correct trading actions. As for the stock market, the generally adopted data is the sequence at regular intervals of time such as the price open, close, high, low (OCHL), and volume. However, as stated before, the raw market data contains a high degree of complexity and noise, which makes it difficult to regress over time even for deep neural networks. To address this problem, technical indicators are preliminarily extracted to summarize the market conditions from different perspectives. Technical indicators used in this research include MA, EMA, MACD, BIAS, VR, and OBV, as reported in Table 1. These technical indicators are heuristic or mathematical calculations based on the price, volume and other data of stocks: MA and EMA stand for average technical indicators; MACD stands for the trend technical indicator; BIAS stands for the overbought and oversold technical indicator; VR stands for the energy-based technical indicator; OBV stands for the volume-based technical indicator.

Moving average (MA) is one of the most basic technical indicators, which is defined as:

$$MA = (pv_0 + pv_1 + \ldots + pv_n)/n \tag{1}$$

where pv represents the price and volume data over a given time period and n represents the time span.

The exponential moving average (EMA) is a trend-oriented indicator aiming at weighting more on recent data than on data from the distant past. It is an exponentially decreasing weighted moving average, which is defined as:

$$EMA_t = \alpha * p_t + (1 - \alpha)EMA_{t-1} \tag{2}$$

where α is the smoothing index, n is the time period, t is the current time, and p_t is the price data at time t.

The moving average convergence divergence ($MACD$) defines the difference between a long-term moving average and a short-term moving average. This indicator not only retains the advantages of moving averages but also avoids producing wrong trading signals. $MACD$ can be formulated as:

$$MACD = 2 * (DIF - DEA)$$

$$DIF = EMA(SHORT) - EMA(LONG) \tag{3}$$

where DEA is the arithmetic mean of DIF over the time span and EMA is defined in Eq. (2).

The $BIAS$ measures the deviation of the market share price from the corresponding moving average during the fluctuation process, which is defined as:

$$BIAS = \frac{p - MA(n,p)}{MA(n,p)} * 100 \tag{4}$$

where p is the stock price and $MA(n;p)$ is the moving average of prices of n time units before.

The volatility-volume ratio (VR) is used to decide whether to buy stocks or sell stocks from the perspective of volume, which is formulated as:

$$VR = \frac{S(n,u) + \frac{1}{2}S(n,v)}{S(n,d) + \frac{1}{2}S(n,v)} * 100 \tag{5}$$

Table 1
The list of used technical indicators.

Technical indicator	Indictor description
MA	Moving Average
EMA	Exponential Moving Average
MACD	Moving Average Convergence/Divergence
BIAS	Bias
VR	Volatility Volume Ratio
OBV	On Balance Volume

where u indicates the volume of the positive line, d indicates the volume of the negative line, v is the volume of flat day, and $S(n,*)$ is the sum of the volume within n days.

The on-balance volume (OBV) estimates the trend of stock price by the statistic of volume changes.

$$OBV_t = OBV_{t-1} + sgn * v_t \tag{6}$$

where t is current time, sgn is signum and v is the volume. When $close_t > close_{t-1}$, sgn is equal to 1, otherwise it is equal to -1.

The above mentioned technical indicators depict the environmental status of stock markets for trading agents. In addition, in order to compare the performance of different trading strategies in various markets, we further define the stock strength (ST) in Eq. (7) to distinguish whether the stock is in a trending market or in a volatile market.

$$ST = \frac{p_n - p_1}{\sum_{i=1}^{n-1} |p_{i+1} - p_1|} \tag{7}$$

where p_n is a price series consisting of past n values. When ST is greater than 0.3, the stock is in a trending market, otherwise it is in a volatile market.

3.2. Trading rules and reward function

The actions of trading agents will vary corresponding to the long, neutral, or short position. The long position represents the buying of a stock or some stocks expecting that it will rise in value. The short position refers to the sale of a stock to another investor by someone who borrows the shares from the investment firm and believes the price of the stock will decrease in value in the short term. The neutral position is to hold on current investment. The trading of stocks should also be related to the quantity exchanged, matching demands, and offers in the market. However, for the sake of simplicity, our experiment is irrespective of this aspect, and it will refer to a nominal quantity of 1 share per trade.

An agent of reinforcement learning methods can learn the optimal policy to trade stocks for the maximum profit. The design of reward function is of vital importance for trading strategies with reinforcement learning methods. In the stock market, it is natural to define the reward function with the rate of return as follows:

$$R = \frac{(SP - BP)}{BP} * 100\% \tag{8}$$

where BP is buying price of the stock, and SP is selling price of the stock. The rate of return is measured by calculating the percentage increase or decrease in the stock's price. However, for the proposed GDQN and GDPG, the reward function r is tailor designed with the Sortino ratio (SR) [18] and defined as follows:

$$\begin{aligned} SR &= \frac{E(R)-T}{D(R)} \\ r &= SR \end{aligned} \tag{9}$$

where R is defined in Eq. (8); $E(R)$ is the average realized return in an episode of trading; T is the risk-free target rate of returns, and $D(R)$ is the target semi-deviation (the square root of target semi-variance), termed as downside deviation. The Sortino ratio can measure the risk-adjusted return of a stock investment strategy, which fits to the highly volatile stock market. It is a modification of the Sharpe ratio [36] and both of them are risk-adjusted evaluations of return on investment. The Sharpe ratio is calculated by subtracting the rate of return on an investment considered risk-free from the expected or actual return on an individual stock, then dividing that number by the standard deviation of the stock, which is good to evaluate low volatile investment. Whereas, the Sortino ratio only focuses on the downside, or negative volatility, rather than the total volatility used in calculating the Sharpe ratio. Because the Sortino ratio only factors in the negative deviation of a trading strategy's returns from the mean, we design the reward function of the proposed GDQN and GDPG with it for a better evaluation of the trading strategy's risk-adjusted performance.

3.3. The state of stock markets

Stock market movements can be depicted by price series and technical indicators, but they cannot reveal the patterns or features behind the evolving states of the market. In order to explore useful patterns and extract informative features, we propose to make use of GRU with deep architecture as shown in Fig. 1.

The GRU contains an update gate and a reset gate, in which the reset gate function is to determine how much previous stock market information can be ignored, and the update gate function is to select whether the hidden state will be updated with the current stock market information.The update gate is used to control the degree to which the market environment information of the previous moment is brought into the current state. The larger the value of the update gate, the more the market environment information is brought in at the previous moment.

$$\begin{aligned} i_t &= \sigma(w_i x_t + u_i h_{t-1} + b_i) \\ x_t &= \{open, close, high, \ldots, VR, OBV\} \end{aligned} \tag{10}$$

where x_t is the input vector at time t, h_{t-1} is the output value of the previous time step, w is input weight matrix, u is the recurrent weight matrices, and b is a bias vector. Furthermore, the σ is the sigmoid function. Besides, the reset gate rg is used to control the degree of ignoring market environment information from the previous moment as follows:

$$rg_t = \sigma(w_r x_t + u_r + b_r) \tag{11}$$

The next step is the calculation of the current hidden state and output.

$$\begin{aligned} h_t^{\sim} &= tanh(w_{h^{\sim}} \cdot [rg_t \cdot h_{t-1}, x_t]) \\ h_t &= (1 - i_t) \cdot h_{t-1} + i_t \cdot h^{\sim} \\ y_t &= \sigma(w_o \cdot h_t) \end{aligned} \tag{12}$$

Based on the time series nature of stock data, the GRU is utilized to construct informative feature representation. In following subsection, the details of GDQN and GDPG will be discussed.

3.4. Critic-only trading strategy: GDQN

A good trading strategy is not to make the most profit out of every single trade, but to make the most profit out of the majority of the trades to ensure a long-term profit in stock markets. The proposed GDQN is a critic-only reinforcement learning method for stock trading and its framework is demonstrated in Fig. 2. To be profitable in the long run, the goal of the GDQN agent is to maximize risk-adjusted returns by interacting with the stock market environment and selecting proper actions. For an episode $t = \{1, \ldots, T\}$, the GDQN will generate a series of transactions with the reward $\{r_1, r_2, r_3, \ldots, r_t\}$, which is defined in Eq. (9). Meanwhile, the G_t is defined as follows to represent the expected returns of GDQN:

$$G_t = r_t + \lambda * r_{t+1} + \lambda^2 * r_{t+2} + \ldots = \sum_{k=1}^{T} \lambda^{k-1} * r_{t+k-1} \tag{13}$$

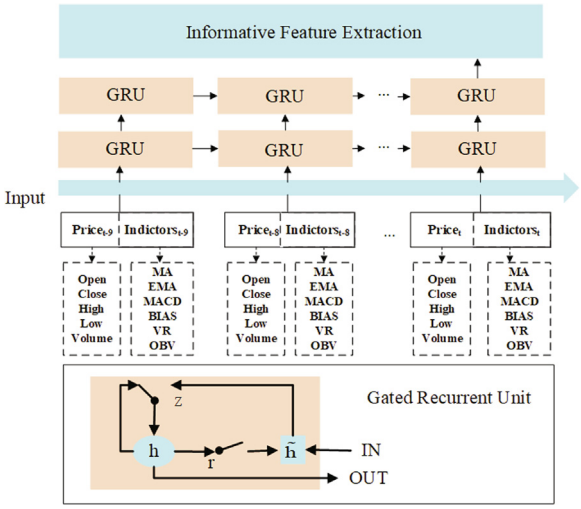


Fig. 1. The deep learning module of the GDQN and GDPG.

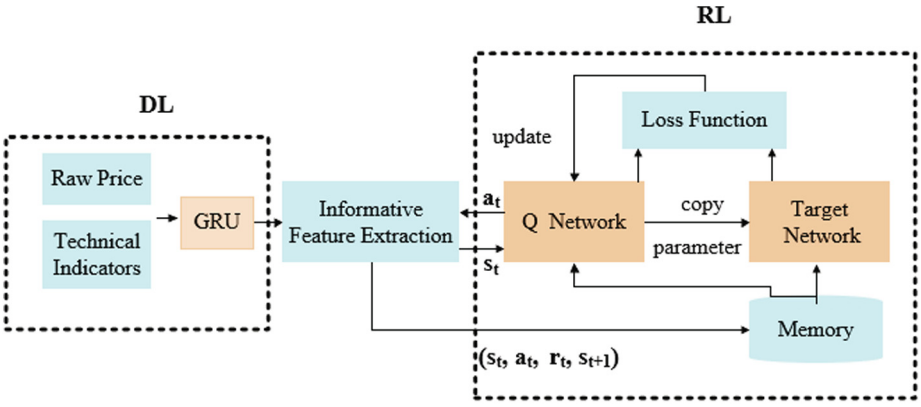


Fig. 2. The framework of GDQN.

where G_t is a sum of discounted rewards after time t, and λ is the discount factor. In another word, G_t is the expected cumulative future discounted rewards. Notice that in Eq. (13), the value of G_t only depends on the current state s_t, and it is, in fact, a property of Markov decision processes. By the law of large numbers, we can estimate the expectation of $G_t(s_t)$ by observing the trading, formally as:

$$v_\pi(s) = E[G_t|S_t = s] \tag{14}$$

Substitute Eq. (14) into Eq. (15) and we have the Bellman equation as follows:

$$\begin{aligned} v(s) &= E[G_t|S_t = s] \\ &= E[r_t + \lambda * (r_{t+1} + \lambda * \ r_{t+2} + \ldots)|S_t = s] \\ &= E[r_t + \lambda * \ G_{t+1}|S_t = s] = E[r_t + \lambda * \ v(S_{t+1})|S_t = s] \end{aligned} \tag{15}$$

Eq. (15) indicts that the value of $v(s_t)$ is determined by r_t and $v(s_{t+1})$. There are multiple actions can be selected by the agent in each state, thus it is better to use $Q(s, a)$ instead of $v(s)$, where $Q(s, a)$ is the expected rewards of the next state of s, given the trading action a. To get as much profit as possible, the proposed GDQN tries to find a strategy that maximizes $v(s)$, or its equivalently $Q(s, a)$. This is defined in Eq. (16) as:

$$\pi^* = argmaxq_\pi(s, a) \tag{16}$$

There are a variety of actions that can be selected by the agent in each market state for the GDQN, and each state is different. It is not enough to calculate the action-value because the optimal strategy is needed. Therefore, based on the valuation iteration, the current Q value and rewards are used to update the historical Q value. The process of updating Q value is defined as:

$$\begin{aligned} &Q = Q(s_t, a_t) + \alpha(R_t + \lambda max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \\ &s_t \leftarrow s_{t+1} \end{aligned} \tag{17}$$

According to Eq. (17), the agent will choose a_t from s_t using a policy derived from Q. After that, the agent will take action a_t, and observe r_t and s_{t+1}. Instead of giving the estimated Q value directly to the new Q value, it will be approached in a gradual way by the learning rata α of GDQN. However, there is a countless amount of states in stock markets, which result in a countless amount of state-action pairs. It is impossible for us to traverse all states to construct an infinite Q-table, which needs infinite memory space. In order to address this issue, the GDQN adds experience replay memory into the training process. The experience replay memory would store each transaction$= (s_t, a_t, r_t, s_{t+1})$ that is produced in the training process. Moreover, the training network samples the random mini-batch empirical data in this experience replay memory, which can avoid the local optimal problem of the learning strategy. To simplify the learning process, the experience replay would make the learning process of the agent similar to supervised learning.

Value function approximation is achieved by transferring the updating problem of Q value matrix to the function fitting problem as follows:

$$Q(s, a, \theta) \approx Q^\pi(s, a) \tag{18}$$

The approximation function is described as a parameterized function of states and actions, which updates the network parameters θ by making the Q function approximates the optimal Q value. In order to increase the random exploration ability of the GDQN agent, the trading actions are randomly selected according to a certain probability. However, it has been proven that the traditional DQN [28] generally overestimates the action-value and the estimation error increases as the number of

trading actions increase, which makes it difficult for agents to find the optimal strategy. To address this problem, two neural networks are created: Q network and target Q network. The Q network is used to select the action a_{t+1} to maximize $Q(s_{t+1}, a_{t+1})$. Meanwhile, the target Q network calculates the action-value of a_{t+1}. The computation process is described as follows:

$$y = r + \lambda Q^{target}(s_{t+1}, argmax_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta); \theta') \tag{19}$$

The selection of actions, in the *argmax*, is closely related to the weights θ of Q network.That is to say, in the GDQN, we estimate the value of the greedy policy according to the current values as defined by θ. However, we use the second set of weights θ' to evaluate the value of this policy. Therefore, it is possible to avoid over-estimated trading actions in stock transactions and the loss function is defined as:

$$L(\theta) = E[(y - Q(s, a; \theta))^2] \tag{20}$$

The target network, with parameters θ', is the same as the online network except that its parameters are copied every τ steps from the Q network. The model structure is shown in Fig. 2.

As shown in Fig. 2, there are two network models: Q network and target network. The Q network selects trading actions based on current market conditions; while the target network calculates the action-value of trading action to update the model. The method reduces overestimations by decomposing the max operation in the target into action selection and action evaluation. The pseudo code of the proposed GDQN algorithm is illustrated in Algorithm 1.

Algorithm 1 The GDQN algorithm

Input: stock environment S_t, Q network and its parameters θ, target Q^{target} network and its parameters θ'
Output: weights θ for Q networks
Initialize the S_t
Initialize the memory replay reposity D to capacity N
Initialize the Q network with random weights θ, and Initial the target Q^{target} network with $\theta' = \theta$
For episode = 1, M do
  For t = 1,T do
    Fetch strock environment from D and form input S_t
    Select $a_t = argmax_{a_t} Q(S_t, a; \theta)$ according to the S_t
    Otherwise select random action a_t with probability ε
    Execute action a_t, reward r_t and calculate S_{t+1}
    Store the transition (S_t, a_t, r_t, S_{t+1}) in D
    Sample minibatch (S_t, a_t, r_t, S_{t+1}) randomly from D
    Set $y = r + \lambda Q^{target}(s_{t+1}, argmax_a Q(s_{t+1}, a_{t+1}; \theta); \theta')$
    Train the network with loss function $L(\theta) = E[(y - Q(s, a; \theta))^2]$
    Reset the Q network $Q^{target} = Q$ every τ steps
  End for
End for
Return weights θ' for Q network

The structure of the Q network is as shown in Fig. 3.

The parameter settings of the Q network of GDQN are shown in Table 2.

3.5. Actor-critic trading strategy: GDPG

The proposed GDQN can be categorized as a critic-only trading strategy. To achieve stable risk-adjusted returns, we propose an actor-critic trading strategy GDPG and its framework is demonstrated in Fig. 4.

The actor module selects trading actions according to the stock market states, which is defined as follows:

$$a = \mu(s_t|\theta^{\mu}) + \varepsilon \tag{21}$$

In Eq. (21), μ stands for the action with the highest probability and ε is random noise added to the selected action. While the critic module will rate the selected trading action as follows:

$$value = Q(s_t, a_t|\theta^{Q}) \tag{22}$$

The correction for action-value at time t is defined as follows:

$$\delta_t = r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \tag{23}$$

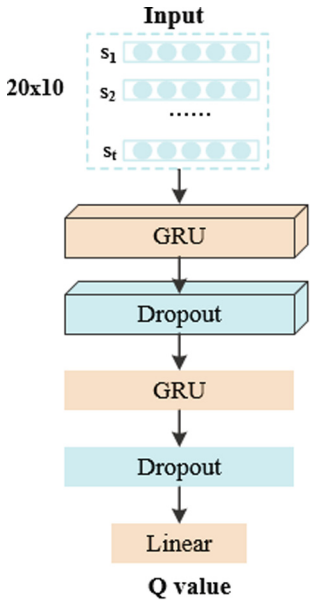


Fig. 3. The network structure of GDQN.

Table 2
The parameter settings of the Q network of GDQN.

Layer	Outout Shape	Parameter
GRU	(10, 32)	5088
Dropout	(10, 32)	0
GRU	32	6240
Dropout	32	0
Dense	3	99

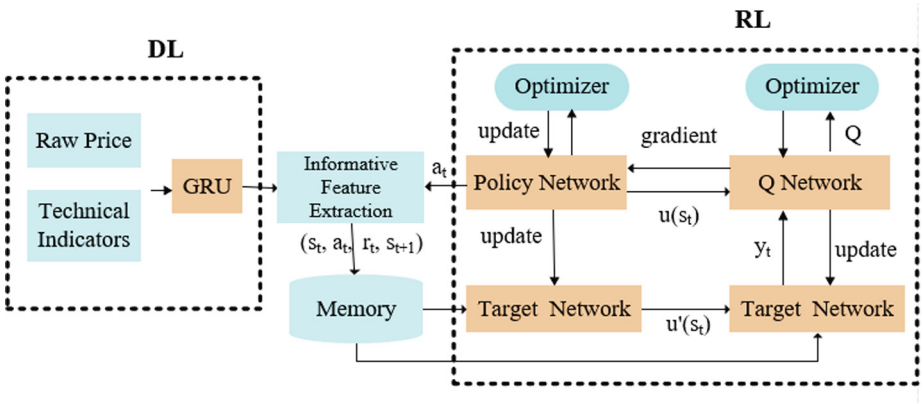


Fig. 4. The framework of GDPG.

The s_t is market environment state at current time t. The *value* in the Eq. (22) estimates the action-value. The actor updates the policy distribution in the direction suggested by the critic.

In the GDPG, both the actor and critic functions are parameterized with neural networks. Due to the strong correlation between the two neural networks in the GDPG, it may not lead to the optimal strategy. To address this problem, target network of the GDPG is tailor designed, which can improve the stability and convergence of the GDPG algorithm.

The proposed GDPG consists of an actor module and a critic module. The actor module is used to update the trading strategy, while the critic module directs the actor module to update the policy distribution. The actor module includes two neural

networks having the same structure but different parameters. At the same time, the critic module also includes two neural networks having the same structure but different parameters. Therefore, the proposed GDPG includes 4 neural networks, and its framework is described in Fig. 4.

The loss function of the critic network is defined as follows:

$$\begin{aligned} Loss &= \frac{1}{N}\sum_{i=1}^{N}(y_i - Q(s_i, a_i|\theta^Q))^2 \\ y_i &= r_i\lambda Q(s_{i+1}, \mu^{target}(s_{i+1}, \theta^{\mu'})|\theta^{Q'}) \end{aligned} \tag{24}$$

In the Eq. (24), the next-state Q value is calculated with the critic target network and target policy network. Then, we minimize the mean-squared loss between the updated Q value and the original Q value. For the policy function, our objective is to maximize the expected return as follows:

$$J(\theta) = E[Q(s, a)|s = s_t, a_t = \mu(s_t)] \tag{25}$$

The mean of the sum of gradients calculated from the mini-batch is defined as follows:

$$\nabla_{\theta^\mu} J(\theta) \approx \frac{1}{N}\sum_i [\nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s=s_i}] \tag{26}$$

The update of parameters of two target networks in the GDPG is based on policy network parameters.

$$\begin{aligned} \theta^Q &= \tau\theta_Q + (1-\tau)\theta^{Q'} \\ \theta^{\mu'} &= \tau\theta_\mu + (1-\tau)\theta^{\mu'} \end{aligned} \tag{27}$$

where τ is the learning rate of GDPG, which is less than 1 and we set its value to 0.001 in the experiment. The pseudo code of the proposed GDPG algorithm is illustrated in Algorithm 2.

Algorithm 2 The GDPG algorithm

Randomly initialize critic network $Q(s_t, a_t|\theta^Q)$ and actor $\mu(s_t|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize memory replay D
For episode = 1, M do
  Initialize a random process ε for action exploration
  Receive initial Stock information state s_1
  For t = 1,T do
    Select action $a = \mu(s_t|\theta^\mu) + \varepsilon$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in D
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from D
    Set $y_i = r_i\lambda Q(s_{i+1}, \mu^{target}(s_{i+1}, \theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss:
      $Loss = \frac{1}{N}\sum_{i=1}^{N}(y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the policy gradient:
    $\nabla_{\theta^\mu} J(\theta) \approx \frac{1}{N}\sum_i [\nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s=s_i}]$
    Update the target networks:
      $\theta^Q = \tau\theta_Q + (1-\tau)\theta^{Q'}$
      $\theta^{\mu'} = \tau\theta_\mu + (1-\tau)\theta^{\mu'}$
  End for
End for

The structure of the actor network and critic network is shown in Fig. 5 and the parameter settings of the actor network and critic network in the GDPG are shown in Table 3.

After the elaboration of proposed GDQN and GDPG, their performances will be evaluated on different stock markets.

4. Experimental results

In order to verify the effectiveness and robustness of the proposed trading strategies, the GDQN and GDPG are evaluated and compared with the Turtle Trading Strategy [29] and a state-of-the-art direct reinforcement learning strategy, DRL trad-

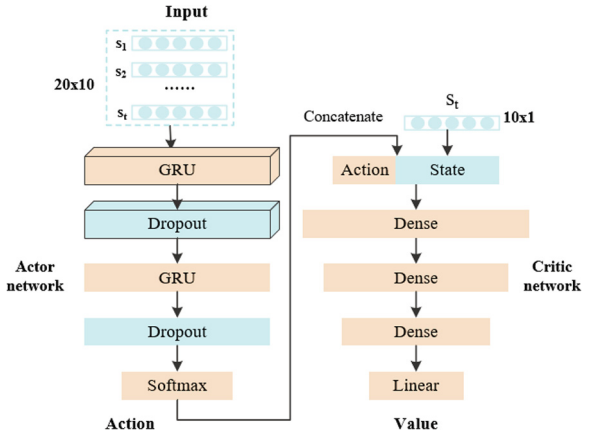


Fig. 5. The network structure of GDPG.

Table 3
The parameter settings of the actor network and critic network of GDPG.

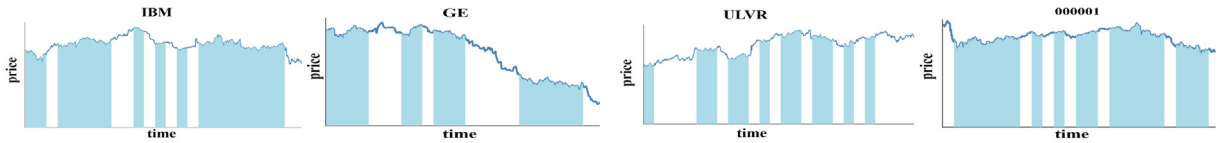
Network	Layer	Output Shape	Parameter
Actor network	GRU	(10,24)	3240
	Dropout	(10,24)	0
	GRU	24	3528
	Dropout	24	0
	Dense	3	75
Critic network	Concatenate	23	0
	Dense	64	1536
	Dense	16	1040
	Dense	1	17

ing strategy [6]. The DRL utilizes an actor-only framework, which learns the policy directly from the continuous sensory data, stock market features, and defines a spectrum of continuous actions according to the learned policy. What's more, the DRL achieved good performance in stock markets. Since the proposed GDQN and GDPG share the same structure as the DRL, it is natural to make comparisons between proposed methods and the DRL trading strategy. The experimental stock data is selected from three countries: the U.S. stock market, the U.K. stock market, and the Chinese stock market. In addition, the selected stocks cover different market conditions: the trending market and the volatile market. The selected stocks are well-known stocks in each country, which are listed in Table 4.

The historical data of selected stocks consists of the raw data and technical indicators. The raw data includes the price open, close, high, low (OCHL) and volume. While technical indicators include MA, EMA, MACD, BIAS, VR, and OBV. Both the raw data and technical indicators are normalized to a 20-dimensional vector to eliminate dimensional inconsistencies. The training dataset is from January 1st, 2008 to December 31st, 2015. The test dataset is from January 1st, 2016 to December 20th, 2018. When the dataset is built, the continuous raw data for about 500 days is used as a learning episode for the agent. Therefore, in one episode, the agent needs to take 200 steps. Each step has three operations: buy, sell, and no operation. The plots of price trend of sampled stocks are shown in Fig. 6.

Table 4
Sample stocks in three stock markets.

Market	Symbol	Company	Symbol	Company
The U.S. stock	AAPL	Apple, Inc.	GE	General Electric
	AXP	American Express	CSCO	Cisco Systems
	IBM	International Business		
The U.K. stock	RDSB	Royal Dutch Shell	GSK	GlaxoSmithKline
	ULVR	Unilever	HSBA	HSBC Holdings PLC
	BATS	British American Tobacco		
The Chinese stock	600519	Kweichow Moutai	000001	PingAn Bank
	601288	Agricultural Bank of China	601988	Bank Of China
	601398	ICBC		



(a) Close price of IBM. (b) Close price of GE. (c) Close price of ULVR. (d) Close price of 000001.

Fig. 6. The close prices of sample stocks.

The stock strength is calculated according to the Eq. (7) and demonstrated in Fig. 6. The blue shaded part indicates that the stock is in a volatile state. It is also interesting to note that four sampled stocks exhibit quite different patterns. Generally, the GE shows a downtrend, while the IBM is in a volatile state most of the time. The ULVR shows a strong upward trend while the 000001 is in a volatile state.

4.1. Experiment setup

All the experiments are carried out on a workstation equipped with an Intel Core i3-8100 Processor, 3.6 GHz x8, 64.0 GB RAM, and GPU GeForce GTX 1050 with 3 GB RAM onboard.

In practice, the proposed GDQN and GDPG trading strategies consist of two parts: market condition summarization and optimal action execution as introduced in Section 3. The proposed GRU senses the dynamic market condition for informative feature extraction. The time step of the feature is set to 10 in this experiment. However, the complication of GRU will result in sampling noise, which may lead to an overfitting problem. Hence, the regularization technique called dropout is utilized to address this issue. The probability of dropout is set to 20 percent. That is to say, we discard 20 percent of neuron input.

As for the execution of the trading actions, the reinforcement learning module is implemented as introduced in Section 3. Stock data between January 1st, 2008 and December 31st, 2015 is collected to train the GDQN, GDPG and DRL trading strategies, while the Turtle trading strategy without neural network needn't be trained. After the GDQN, GDPG, and DRL trading strategies are well trained, all trading strategies are evaluated on stock data between January 1st, 2016 to December 20th, 2018. For all trading strategies, the trading capital is the same, which is set at 10 times the historic peak price of each stock during the trading period.

These strategies are evaluated from two aspects: the rate of return (R) defined in Eq. (8) and the Sortino ratio (SR). The Sortino ratio is a widely-used measurement of risk-adjusted return, which is defined in Eq. (9).

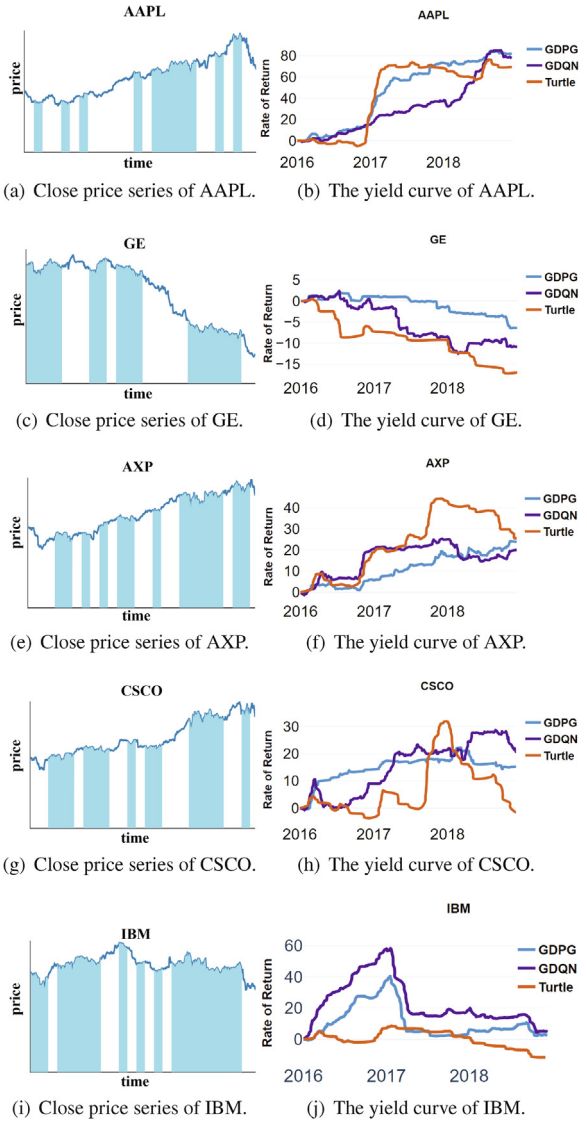
4.2. The comparison of GDQN, GDPG and Turtle trading strategies

In this section, the experimental results are presented. The GDQN and the GDPG are evaluated and compared with the Turtle trading Strategy. The results of GDQN, GDPG, and Turtle trading strategies in the U.S. stock market are demonstrated in Table 5. In addition, the performance of GDQN, GDPG, and Turtle trading strategies are compared in different stock market conditions. The cumulative yield curve of three strategies in the volatile market and the trending market are shown in Fig. 7.

As shown in Table 5 and Fig. 7, when the stock price increases significantly such as that of AAPL and AXP, all three strategies have good returns, but the proposed GDPG has the best performance in APPL and the Turtle trading strategy has the best performance in AXP. When the stock price fluctuates sharply, such as that of IBM, the proposed GDQN and GDPG trading strategies achieve small profits, while the Turtle trading strategy suffers a loss. When stock prices are in a downtrend, all three strategies suffer loses, but the proposed GDQN and GDPG have better performance than the Turtle trading strategy. It is clearly demonstrated from the experimental results, three strategies achieve profitability on some stocks especially those stocks in an upward trend. The Turtle trading strategy only outperforms the proposed GDQN and GDPG in the stock in a clear upward trend. For stocks in a downtrend such as GE, all three tradiing strategies suffer losses because GE had been falling for a long period. However, in the volatile market, only the proposed GDQN and GDPG trading strategy achieve profitability.

Table 5
The GDQN, GDPG and Turtle Strategies in the U.S. stock market.

	GDQN		GDPG		Turtle	
Symbol	SR	R(%)	SR	R(%)	SR	R(%)
AAPL	1.02	77.7	1.30	**82.0**	1.49	69.5
GE	−0.13	−10.8	-0.22	**−6.39**	−0.64	−17.0
AXP	0.39	20.0	0.51	24.3	0.67	**25.6**
CSCO	0.31	**20.6**	0.57	13.6	0.12	−1.41
IBM	0.07	**4.63**	0.05	2.55	−0.29	-11.7



(a) Close price series of AAPL. (b) The yield curve of AAPL.

(c) Close price series of GE. (d) The yield curve of GE.

(e) Close price series of AXP. (f) The yield curve of AXP.

(g) Close price series of CSCO. (h) The yield curve of CSCO.

(i) Close price series of IBM. (j) The yield curve of IBM.

Fig. 7. The comparison of trading strategies in the U.S. stock market.

The results of GDQN, GDPG and Turtle trading strategies in the U.K. stock market are demonstrated in Table 6. The cumulative yield curve of these trading strategies in the U.K. stock market are shown in Fig. 8. As shown in Fig. 8, the stock prices of HSBA and RDSB are in an upward trend, and thus the GDQN, GDPG and Turtle trading strategies can achieve high returns. The stock price of ULVR is in strong volatility, and thus Turtle strategy suffers losses, but the GDQN and GDPG strategies still achieve good returns. The stock price of GSK is in a constant fluctuation, and thus all three strategies fail to achieve satisfac-

Table 6
The GDQN, GDPG and Turtle Strategies in the U.K. stock market.

	GDQN		GDPG		Turtle	
Symbol	SR	R (%)	SR	R(%)	SR	R(%)
RDSB	0.35	**26.8**	0.79	22.7	0.36	10.9
GSK	-0.04	−5.2	0.06	**1.10**	-0.26	-9.9
ULVR	0.41	**21.0**	0.21	11.1	-0.16	-7.07
HSBA	0.76	**46.0**	0.99	45.2	0.90	24.8
BATS	0.14	19.1	0.03	**37.0**	0.51	18.5

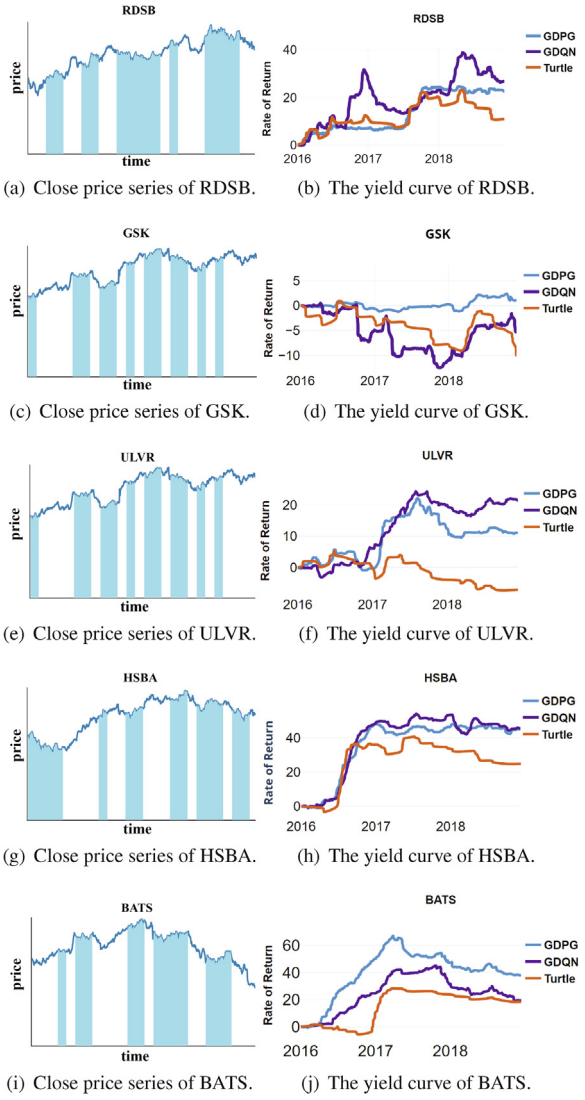


Fig. 8. The comparison of trading strategies in the U.K. stock market.

tory returns while the GDPG still makes profits. The results of GDQN, GDPG and Turtle trading strategies in the Chinese stock market are demonstrated in Table 7.

The performance of GDQN, GDPG and Turtle trading strategies are compared in different stock market conditions. The cumulative yield curve of these strategies in the volatile market and trending market are shown in Fig. 9. Similar to U.S. and U.K. stock market, when the stock price is in an upward trend, all three trading strategies have achieved good profitability. When the stock is in a volatile state, such as 00001, the three strategies suffer loses, but the proposed GDQN and GDPG perform better than the Turtle trading strategy. From the experimental results, we can safely draw a conclusion that the proposed GDQN and GDPG trading strategies perform better than the Turtle trading strategy not only in the trending market but also in the volatile market, which has been proven by the performance of GDQN, GDPG and Turtle trading strategies in the U.S., the U.K. and the Chines stock markets.

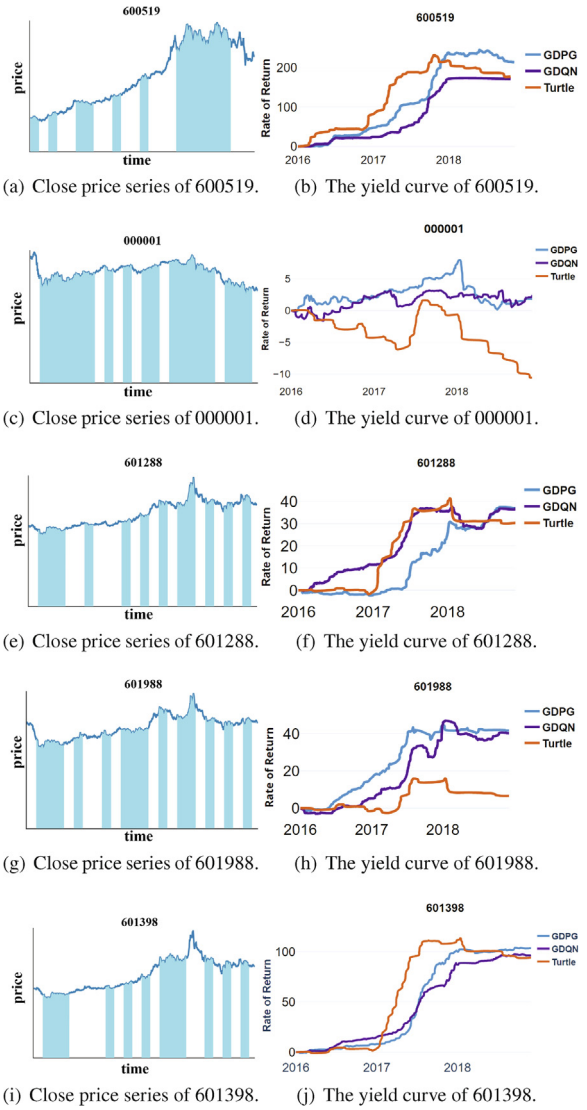
4.3. The comparison of GDQN, GDPG and DRL trading strategies

To further evaluate the performance of the proposed GDQN and GDPG strategies, they are compared with a state-of-the-art direct reinforcement learning (DRL) strategy for stock trading. The Sortino ratio and rate of return are also used to evaluate these trading strategies' performance. The comparison of GDQN, GDPG and DRL strategies are demonstrated in Table 8.

The Table 8 shows the corresponding yield and Sortino ratio of GDQN, GDPG and DRL strategies in the U.S., the U.K., and the Chinese stock markets. From Table 6, it is clearly demonstrated that the DRL trading strategy does achieve profitability on

Table 7
The GDQN, GDPG and Turtle Strategies in the Chinese stock market.

	GDQN		GDPG		Turtle	
Symbol	SR	R (%)	SR	R(%)	SR	R(%)
600519	1.79	171.0	1.90	**215.9**	0.4	15.1
000001	0.07	**2.46**	0.07	1.99	-0.37	-10.5
601288	0.86	36.8	0.96	**37.1**	1.20	30.4
601988	0.62	40.1	1.01	**41.6**	0.30	6.62
601398	3.47	96.6	2.60	**103.3**	2.45	94.1



(a) Close price series of 600519. (b) The yield curve of 600519.

(c) Close price series of 000001. (d) The yield curve of 000001.

(e) Close price series of 601288. (f) The yield curve of 601288.

(g) Close price series of 601988. (h) The yield curve of 601988.

(i) Close price series of 601398. (j) The yield curve of 601398.

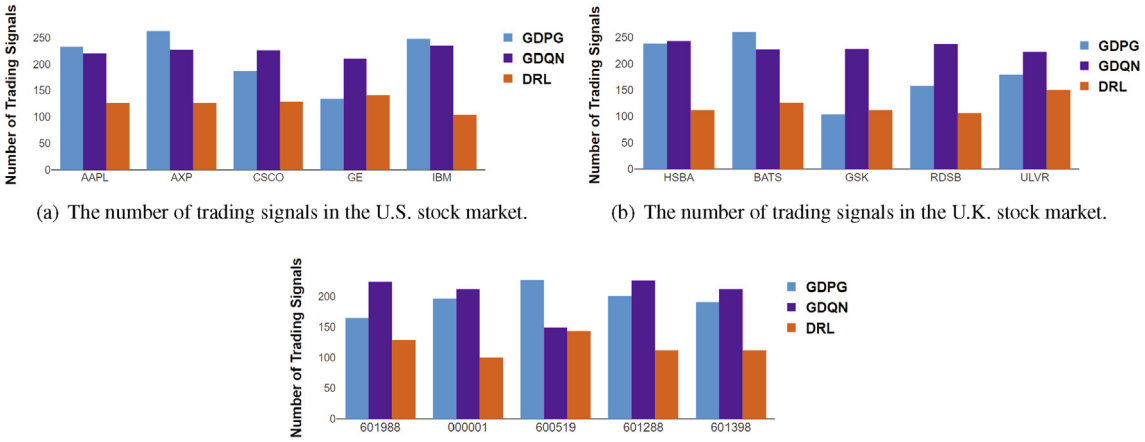
Fig. 9. The comparison of trading strategies in the Chinese stock market.

most stocks. While the DRL suffers losses on GE and IBM in the U.S. stock market, GSK in the U.K. stock market and 000001 in the Chinese stock market, which has less stable returns than the proposed GDQN and GDPG in volatile market conditions.

Furthermore, the number of trading signals generated by the GDQN, GDPG and DRL strategies in different stock markets are also plotted in Fig. 10. For trading strategies with reinforcement learning methods, the number of trading signals means the sensitivity to market changes. It can be seen from Fig. 10 that the total number of trading signals generated by the DRL trading strategy in the three markets is significantly less than that of the proposed GDQN and GDPG, which means the DRL is less sensitive to market changes than the proposed GDQN and GDPG.

Table 8
The GDQN, GDPG and DRL Strategies in three stock markets.

	GDQN		GDPG		DRL	
Symbol	SR	R (%)	SR	R(%)	SR	R(%)
GE	−0.13	−10.8	-0.22	**−6.39**	−0.24	−10.1
AAPL	1.02	77.7	1.30	**82.0**	0.25	15.8
AXP	0.39	20.0	0.51	**24.3**	0.30	15.2
CSCO	0.31	**20.6**	0.57	13.6	0.14	7.8
IBM	0.07	**4.63**	0.05	2.55	−0.29	-3.12
RDSB	0.35	**26.8**	0.79	22.7	0.16	7.56
GSK	-0.04	−5.2	0.06	**1.10**	-0.18	-8.82
ULVR	0.41	**21.0**	0.21	11.1	0.19	13.1
HSBA	0.76	**46.0**	0.99	45.2	0.03	0.99
BATS	0.14	19.1	0.03	**37.0**	0.16	6.75
600519	1.79	171.0	1.90	**215.9**	0.30	33.3
000001	0.07	**2.46**	0.07	1.99	-0.14	-6.69
601288	0.86	36.8	0.96	**37.1**	0.23	12.5
601988	0.62	40.1	1.01	**41.6**	0.27	12.1
601398	3.47	96.6	2.60	**103.3**	0.27	7.02



(a) The number of trading signals in the U.S. stock market.

(b) The number of trading signals in the U.K. stock market.

(c) The number of trading signals in the Chinese stock market.

Fig. 10. The comparison of the number of trading signals in three stock markets.

As far as the GDQN and GDPG are compared, they achieve good profitability on most stocks. While the GDPG has higher Sortino Ratio than the GDQN in some stocks, which can be demonstrated in their performance in AAPL, AXP and CSCO from the U.S. stock market, RDSB, GSK, HSBA from the U.K. stock market, and 600519, 601288, 601088 from the Chinese stock market. What's more, the GDPG's loss is much smaller than that of the GDQN in a stock in a downtrend such as GE from the U.S. stock market and GSK from the U.K. stock market. At the same time, the GDPG's yield curve is more stable than the GDQN's yield curve as demonstrated in Fig. 7, Fig. 8, and Fig. 9. To sum up, the GDPG is more stable than the GDQN in the ever-evolving stock market.

All trading strategies are evaluated in the U.S., the U.K., and the Chinese stock markets both in the trending and in the volatile stock market conditions. Experimental results show that the proposed GDQN and GDPG have better performance than the Turtle trading strategy and the DRL trading strategy in the Sortino ratio and rate of return. Furthermore, the GDPG is even more stable than the GDQN according to the evaluating indicator Sortino Ratio.

5. Discussion

Deep reinforcement learning in stock trading is new horizons not only in industry but also in academia. Stock trading is the buying and selling of shares of one or some companies. A quantitative stock trading strategy relies on quantitative analysis, which combines mathematical computations with statistical technical indicators to identify market patterns and make trading actions. While deep reinforcement learning is to take suitable actions to maximize reward in a particular situation. Thus it is reasonable for reinforcement learning methods to play some roles in quantitative stock trading. Consequently, the GDQN and the GDPG are proposed to maximize the profitability through a sequence of steps in different stock markets.

There is a potential for the proposed GDQN and GDPG in quantitative stock trading due to the following characteristics of stock markets:

1. The size of the stock markets is large and it requires quantitative description of extensive continuous data.
2. Trading actions may result in long-term consequences and it cannot be directly measured by other supervised learning techniques.
3. All trading actions also have short-term effects on current stock market conditions which make the market unpredictable.

The proposed GDQN and GDPG are the combination of deep neural networks and reinforcement learning and they are promising in quantitative stock trading due to following advantages:

1. They extract informative financial features with GRU from raw financial data and technical indicators to improve the accuracy and robustness for the representation of stock market conditions.
2. Their self-learning ability suits the ever-changing market environment.
3. They are effective and efficient in a high-density environment.

From the experimental results, it can safely draw a conclusion that the proposed GDQN and GDPG outperform the Turtle trading strategy and the DRL trading strategy not only in trending but also in volatile stock market.

In stock markets, the actor-only reinforcement learning method takes as input the states of market environment and outputs the optimal to select actions, which drives the agent to select actions by learning the optimal policy. Whereas the critic-only reinforcement learning method, on the other hand, evaluates actions by computing the value function. The actor-critic reinforcement learning method aims to be benefit to all the good stuff from both actor-only and critic-only methods while eliminating their drawbacks. The actor-critic reinforcement learning method has two models simultaneously: one is to compute an action based on a market state and the other one is to produce the Q values of the action. Thus the proposed GDPG with an actor-critic framework is proven to be able to learn big, complex stock market environments and has more stable returns than the GDQN with a critic-only framework.

6. Conclusion and future work

In order to make profit from the stock market, the GDQN and the GDPG are proposed for quantitative stock trading. The proposed methods are composed of deep learning module and reinforcement learning module. The deep learning module is used to extract stock market features from raw market data and technical indicators. For the time-series nature of financial data, the deep learning module is implemented with the GRU. Meanwhile, reinforcement learning module executes trading actions through GRU extracted deep features. Extensive experiments are carried out to verify the effectiveness and robustness of the proposed GDQN and GDPG methods. According to the experimental results, the proposed GDQN and GDPG are effective in diverse stock markets. They overcome the shortcomings of traditional trading strategies that can only perform well in a single market pattern. Furthermore, compared with the state-of-the-art reinforcement learning method, DRL trading strategy, our proposed methods effectively capture more market opportunities to achieve more stable returns under an acceptable risk level.

Although the proposed GDQN and GDPG trading strategies have good performance in different stock markets, they haven't included the idea of portfolio management. Investment in a single stock has a limited profit margin and comparatively high risk. A good quantitative stock trading strategy needs to build portfolios of multiple stocks. Hence, our future work will focus on utilizing deep reinforcement learning methods in the portfolio management.

CRediT authorship contribution statement

Xing Wu: Conceptualization, Methodology, Software. **Haolei Chen:** Software, Writing - original draft. **Jianjia Wang:** Data curation, Validation. **Luigi Troiano:** Writing - review & editing. **Vincenzo Loia:** Supervision. **Hamido Fujita:** Supervision, Conceptualization, Writing - review & editing.

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 R&D Program of China under Grant 2019YFE0190500 and by the State Key Program of National Nature Science Foundation of China (Grant No. 61936001).

References

[1] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen, et al, Deep speech 2: End-to-end speech recognition in english and mandarin, in: International conference on machine learning, 2016, pp. 173–182.
[2] W. Bao, J. Yue, Y. Rao, A deep learning framework for financial time series using stacked autoencoders and long-short term memory, PloS one 12 (2017) e0180944.
[3] J. Carapuço, R. Neves, N. Horta, Reinforcement learning applied to forex trading, Appl. Soft Comput. 73 (2018) 783–794.
[4] Z. Che, S. Purushotham, K. Cho, D. Sontag, Y. Liu, Recurrent neural networks for multivariate time series with missing values, Sci. Rep. 8 (2018) 6085.
[5] S. Chinchali, P. Hu, T. Chu, M. Sharma, M. Bansal, R. Misra, M. Pavone, S. Katti, Cellular network traffic scheduling with deep reinforcement learning, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018, pp. 766–774.
[6] 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 (2017) 653–664.
[7] Y. Deng, Y. Kong, F. Bao, Q. Dai, Sparse coding-inspired optimal trading system for hft industry, IEEE Trans. Ind. Inform. 11 (2015) 467–475.
[8] T. Fischer, C. Krauss, Deep learning with long short-term memory networks for financial market predictions, Eur. J. Oper. Res 270 (2018) 654–669.
[9] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
[10] B. Huang, Y. Huan, L.D. Xu, L. Zheng, Z. Zou, Automated trading systems statistical and machine learning methods and hardware implementation: a survey, Enterprise Inform. Syst. 13 (2019) 132–144.
[11] H.D. Huynh, L.M. Dang, D. Duong, A new model for stock price movements prediction using deep neural network, in: Proceedings of the Eighth International Symposium on Information and Communication Technology, ACM, 2017, pp. 57–62.
[12] A. Kumar, O. Irsoy, P. Ondruska, M. Iyyer, J. Bradbury, I. Gulrajani, V. Zhong, R. Paulus, R. Socher, Ask me anything: Dynamic memory networks for natural language processing, International conference on machine learning, 2016, pp. 1378–1387.
[13] G. Lample, D.S. Chaplot, Playing fps games with deep reinforcement learning, Thirty-First AAAI Conference on, Artif. Intell. (2017) 2140–2146.
[14] S. Li, Y. Wu, X. Cui, H. Dong, F. Fang, S. Russell, Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient, AAAI Conference on Artificial Intelligence (AAAI), 2019, pp. 4213–4220.
[15] R. Lowe, Y. Wu, A. Tamar, J. Harb, O.P. Abbeel, I. Mordatch, Multi-agent actor-critic for mixed cooperative-competitive environments, Adv. Neural Inform. Process. Syst. (2017) 6379–6390.
[16] B. Luo, D. Liu, H.N. Wu, D. Wang, F.L. Lewis, Policy gradient adaptive dynamic programming for data-based optimal control, IEEE Trans. Cybern. 47 (2017) 3341–3354.
[17] C. Martinez, E. Ramasso, G. Perrin, M. Rombaut, Adaptive early classification of temporal sequences using deep reinforcement learning, Knowledge-Based Syst. 190 (2019).
[18] V. Mohan, J.G. Singh, W. Ongsakul, Sortino ratio based portfolio optimization considering evs and renewable energy in microgrid power market, IEEE Trans. Sustainable Energy 8 (2016) 219–229.
[19] J.E. Moody, M. Saffell, Y. Liao, L. Wu, Reinforcement learning for trading systems and portfolios, KDD, 1998, pp. 279–283.
[20] M. Pei, X. Wu, Y. Guo, H. Fujita, Small bowel motility assessment based on fully convolutional networks and long short-term memory, Knowl.-Based Syst. 121 (2017) 163–172.
[21] D.S.S. Pinto, K.R.G. da Silva, Robot position control in pipes using q learning, in: 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC), IEEE, 2016, pp. 004609–004613.
[22] M. Qiu, Y. Song, F. Akagi, Application of artificial neural network for the prediction of stock market returns: the case of the japanese stock market, Chaos, Solitons Fractals 85 (2016) 1–7.
[23] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al, A general reinforcement learning algorithm that masters chess, shogi, and go through self-play, Science 362 (2018) 1140–1144.
[24] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al, Mastering the game of go without human knowledge, Nature 550 (2017) 354.
[25] J. Sun, H. Fujita, P. Chen, H. Li, Dynamic financial distress prediction with concept drift based on time weighting combined with adaboost support vector machine ensemble, Knowl.-Based Syst. 120 (2017) 4–14.
[26] J. Sun, H. Li, H. Fujita, B. Fu, W. Ai, Class-imbalanced dynamic financial distress prediction based on adaboost-svm ensemble combined with smote and time weighting, Inform. Fusion 54 (2020) 128–144.
[27] L. Troiano, E.M. Villa, V. Loia, Replicating a trading strategy by means of lstm for financial industry applications, IEEE Trans. Ind. Inform. 14 (2018) 3226–3234.
[28] H. Van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double q-learning, in: Thirtieth AAAI Conference on Artificial Intelligence, 2016, pp. 2094–2100.
[29] D. Vezeris, I. Karkanis, T. Kyrgos, Adturtle: an advanced turtle trading system, J. Risk Financial Manage. 12 (2019).
[30] H. Wang, Y. Wu, G. Min, J. Xu, P. Tang, Data-driven dynamic resource scheduling for network slicing: a deep reinforcement learning approach, Inf. Sci. 498 (2019) 106–116.
[31] P. Wolf, C. Hubschneider, M. Weber, A. Bauer, J. Härtl, F. Dürr, J.M. Zöllner, Learning how to drive in a real world simulation with deep q-networks, 2017 IEEE Intelligent Vehicles Symposium (IV), IEEE, 2017, pp. 244–250.
[32] M. Wolter, A. Yao, Complex gated recurrent neural networks, Adv. Neural Inform. Process. Syst. (2018) 10536–10546.
[33] X. Wu, H. Chen, C. Chen, M. Zhong, S. Xie, Y. Guo, H. Fujita, The autonomous navigation and obstacle avoidance for usvs with anoa deep reinforcement learning method, Knowl.-Based Syst. 196 (2020) 105201.
[34] X. Wu, M. Zhong, Y. Guo, H. Fujita, The assessment of small bowel motility with attentive deformable neural network, Inf. Sci. 508 (2020) 22–32.
[35] T. Xu, S. Zou, Y. Liang, Two time-scale off-policy td learning: Non-asymptotic analysis over markovian samples, Advances in Neural Information Processing Systems, vol. 32, Curran Associates Inc, 2019, pp. 10634–10644.
[36] I. Yevseyeva, E.B. Lenselink, A. de Vries, A.P. IJzerman, A.H. Deutz, M.T. Emmerich, Application of portfolio optimization to drug discovery, Inform. Sci. 475 (2019) 29–43.
[37] T. Young, D. Hazarika, S. Poria, E. Cambria, Recent trends in deep learning based natural language processing, IEEE Comput. Intell. Mag. 13 (2018) 55–75.
[38] S. Yu, Z. Li, Forecasting stock price index volatility with lstm deep neural network, in: Recent Developments in Data Science and Business Analytics, Springer, 2018, pp. 265–272.
[39] Y. Yuan, Z.L. Yu, Z. Gu, Y. Yeboah, W. Wei, X. Deng, J. Li, Y. Li, A novel multi-step q-learning method to improve data efficiency for deep reinforcement learning, Knowl.-Based Syst. 175 (2019) 107–117.
[40] H. Zheng, J. Fu, T. Mei, J. Luo, Learning multi-attention convolutional neural network for fine-grained image recognition, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 5209–5217.
[41] X. Zhong, D. Enke, Forecasting daily stock market return using dimensionality reduction, Expert Syst. Appl. 67 (2017) 126–139.
[42] W. Zhu, C. Lan, J. Xing, W. Zeng, Y. Li, L. Shen, X. Xie, Co-occurrence feature learning for skeleton based action recognition using regularized deep lstm networks, in: Thirtieth AAAI Conference on Artificial Intelligence, 2016, pp. 3697–3703.