Hindawi
Scientific Programming
Volume 2022, Article ID 5812546, 9 pages
https://doi.org/10.1155/2022/5812546

Research Article

Deep Reinforcement Learning for Stock Prediction

Junhao Zhang[1] **and Yifei Lei**[2]

[1]*Zhejiang University, School of Economics, Zhejiang, China*
[2]*Zhejiang University, School of the Public Affairs, Zhejiang, China*

Correspondence should be addressed to Junhao Zhang; 12101010@zju.edu.cn

Received 11 December 2021; Revised 21 January 2022; Accepted 25 January 2022; Published 30 April 2022

Academic Editor: Punit Gupta

Investors are frequently concerned with the potential return from changes in a company's stock price. However, stock price fluctuations are frequently highly nonlinear and nonstationary, rendering them to be uncontrollable and the primary reason why the majority of investors earn low long-term returns. Historically, people have always simulated and predicted using classic econometric models and simple machine learning models. In recent years, an increasing amount of research has been conducted using more complex machine learning and deep learning methods to forecast stock prices, and their research reports also indicate that their prediction accuracy is gradually improving. While the prediction results and accuracy of these models improve over time, their adaptability in a volatile market environment is questioned. Highly optimized machine learning algorithms include the following: FNN and the RNN are incapable of predicting the stock price of random walks and their results are frequently not consistent with stock price movements. The purpose of this article is to increase the accuracy and speed of stock price volatility prediction by incorporating the PG method's deep reinforcement learning model. Finally, our tests demonstrate that the new algorithm's prediction accuracy and reward convergence speed are significantly higher than those of the traditional DRL algorithm. As a result, the new algorithm is more adaptable to fluctuating market conditions.

1. Introduction

One of the most common concerns of financial analysts and investors is making accurate predictions about stock prices [1], but it is also a difficult task for them [2]. This is because the majority of stock prices are highly volatile. Numerous different types of factors influence stock prices. Direct economic indicators such as fluctuations in consumer supply and demand and commodity price indexes are complex enough. Stock price changes have become increasingly elusive as a result of the global environment and investor's behavior [3]. Each factor generates forces in distinct directions, which eventually result in a change in the stock price. The correlation between various factors is frequently so obscure that it is difficult to identify the factors affecting stock prices, let alone build a model for stock price prediction on this basis.

According to today's popular value investment theory, the true value of a stock is usually determined by the market value of the company that issued it. Occasionally, however, stock prices deviate from rational market expectations. The volatility and dynamics of the stock price directly contradict the statistical model's and foundation's assumptions, frequently resulting in inaccurate or even negative prediction results [4, 5].

The well-known efficient market hypothesis argues that it is impossible to forecast the value of stocks and that their movement is random, effectively invalidating numerous attempts to forecast stock prices using historical data. However, a ten-year technical analysis reveals that the values of the majority of stocks are contained within their historical stock prices; thus, collecting and analyzing historical stock price movements is critical for forecasting future stock prices [6].

Machine learning (ML) is a highly effective technique that enables machines to learn autonomously via algorithms. Academia generally believes that machine learning has a strong ability to identify useful data and generalize patterns [7]. The advent of machine learning has resolved one of the

most perplexing problems in stock price prediction: nonlinearity. Machine learning is particularly adept at detecting nonlinear patterns in data. Additionally, deep learning (DL), a method for extracting critical information from nonlinear time series that uses a multilevel network structure, performs even better.

Recent advances in machine learning and deep learning have enabled more accurate stock forecasting, and the majority of papers and studies have demonstrated that their models can outperform the market average or generate quite high returns. Certain fundamental machine learning (ML) models, such as feedforward neural networks (FFNN) and deep learning models, such as recurrent neural networks (RNN) with memory structure and dynamics, are, however, incapable of forecasting data with unstable time series and long-term autoregression [8].

Reinforcement learning (RL) is a subfield of deep learning that is distinct from other fields such as statistical data analysis and supervised learning. It is a strategy that seeks to maximize profits while adapting constantly to changes in the environment in which it operates. Supervised learning's objective is to condense the mapping from input to output, which frequently results in the omission of critical information. Due to the frequency and duration of stock price fluctuations, reinforcement learning (RL) is a more suitable predictive tool for statistical analysis of data than supervised learning.

RL can be classified into five categories based on its sample efficiency: (1) model-based, (2) off-policy, (3) actor-critic, (4) on-policy, and (5) evolutionary gradient-free. In this case, 1 is the most efficient sample size option, while 5 is the least efficient. Among them, model-based and non-policy methods, such as TD-learning [9] and *Q*-learning [10], are referred to as "the critic-only methods" [11] and are used to solve discrete area optimization problems. Luo et al. pioneered the use of the TD algorithm [12]. She believes that the stock price prediction problem can be optimized using reinforcement learning algorithms as a Markov process. She used a reinforcement learning algorithm called TD (0), which only learns through experience, to determine the state value of each state that corresponds to the current stock price trend.

On-policy and evolutionary gradient-free methods are often referred to as "the actor-only method" [13]. One of its applications is policy gradient [14], which can learn parameterized strategies via continuity. The middle section, dubbed "the actor-critic method," [15], is a synthesis of the two methods discussed previously. It is capable of optimizing the ultimate return while also calculating the parameterized strategy and adhering to the value creation of benefits principle.

In RL, the agent chooses which action to take in order to maximize his or her reward. This means that throughout the learning process, the agent maximizes the accumulated rewards and thus develops optimal strategies for a variety of problems. Now, researchers can apply reinforcement learning to a variety of interesting projects, including solving the Rubik's cube [16], improving unmanned driving [17], and Atari and first-person shooter game play [18, 19].

There have been many studies using reinforcement learning to predict stock prices and construct trading models. Lee et al. expanded the research field from simple stock forecasting to a trading system with risk management capabilities [8] and after that they changed the original single-agent research into a multiagent research, which increased the complexity of the system and more adapted to the research method of stocks in 2007 [1].

Other researchers approach the problem from the standpoint of raw data input. Moody et al. and others use financial data from the company as a quantitative basis [20]. Yue et al. proposed an optimal trading system inspired by sparse coding that is well suited for real-time high-frequency trading [21]. This method significantly improves the outcome of raw data selection.

However, the previous study did not completely extract useful financial data due to the noise and fluctuations inherent in a fluctuating market. This article will analyze and predict data using two machine learning frameworks: "critic-only DQN" and "actor-critic deep PG." DQN converts a single-layer network to a convolutional network with multiple layers. It not only enables experience replay but also enables our network to self-train using its memorized history, which more closely matches the time autocorrelation and intertemporal correlation that have emerged in the American market. The PG gradient descent algorithm is a superior strategy to the "Greedy Policy." It determines the rising direction of the objective function by computing its gradient and then adjusts the probability of the action based on its performance to maximize the return. Due to the stock market's relatively weak market power, it is frequently vulnerable to other factors. The optimal strategy based on greedy concepts frequently requires a significant amount of time to learn and is prone to overload problems. After implementing PG, the machine's load can be significantly reduced, and the time required for optimization strategies can be significantly reduced.

2. Fundamentals of Reinforcement Learning

In the context of reinforcement learning, an agent obtains information from the environment. The agent adapts the received data to the environment's current state. The AI then decides on an action to be taken based on the rewards associated with each choice. Moreover, each action alters the environment and the total number of reward points. Reinforcements for rewarding or punishing specific actions are immediately added on the basis of the new state. This interaction between action and environment will persist until the agent masters the art of choosing a decision strategy that maximizes the total return.

The terms above, rewards and environment, refer to two of the four critical factors affecting the RL problem as described by Sutton and Barto. Policies are similar to the rules we impose on ourselves when we operate in the environment. The reward function serves as the overarching goal of training and serves as the standard against which all other factors are measured. A value function specifies the value of a state or state-action pair, which indicates the long-run goodness of the state or state-action pair.

The flowchart is shown in Figure 1. The flowchart of RL illustrates the RL procedure for agent-environment interaction at a high level. In one process, the agent performs an action, in reply to the current state. Then, as a result of this action, the system receives a signal, a reward, r_{t+1}, to direct the behavior of the action in the subsequent time step and to adjust the state in the subsequent time step, using the probability function $f(a_t, s_t)$.

2.1. Agent-Environment Interaction in Reinforcement Learning. The goal of RL is to maximize the agent's reward in a limited number of actions by mapping environmental states to actions. The Markov decision process (MDP) is a mathematical model for the RL problem. Quad (S, A, ρ, f) often expressed as a state of MDP.

In this case, S denotes the collection of all environmental states. $s_t \in S$ denotes the current state in the collection of all environmental states at time t; the set of possible actions taken by the agent is denoted by the letter A. $a_t \in A$ denotes the agent's action at time t; $\rho: S \times A \longrightarrow R$ is the reward function. $r_t \sim \rho(s_t, a_t)$ is a reward and punishment table. It summarizes all of the benefits that a single possible action, a_t, can obtain in a given state, s_t. $f: S \times A \times S \longrightarrow [0, 1]$ is the probability distribution function for state transitions. The probability of state s_t successfully transitioning to state, s_{t+1}., is denoted by $s_{t+1} \sim f(s_t, a_t)$.

In reinforcement learning, the ultimate goal is to enable the agent to discover an optimal action mode, and thus, the strategy we use during this process is critical. The strategy, π: $\mathbf{S} \longrightarrow \mathbf{A}$, indicates that a_t is the optimal action step determined by the current environmental conditions s_t, that is, $\pi(s_t) = a_t$. Assuming that the immediate rewards obtained at each future period should be multiplied by a discount factor in order to avoid infinite returns and to discount future value into the present, time T denotes the plot's conclusion. The following here is an example of a typical function:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{T-1} \gamma^k r_{t+k+1}. \tag{1}$$

Here, γ $(0 < \gamma < 1)$ denotes a constant discount rate for future and current rewards. The strategy is followed until the training is complete while equation (2) refers to action a in the current state s. In another way, it shows a state action value function. The agent's cumulative return during this procedure is stated as follows:

$$Q^{\pi}(s, a) = E\left[R_t | s_{t=s}, a_{t=a}, \pi\right]. \tag{2}$$

If there is a strategy π^*, with an expected return greater than or equal to that of any other strategy for all state action pairs, we refer to it as the optimal strategy. For the strategy, which is called the optimal state action value function, the formula is as follows:

$$Q^{\pi}(s, a) = \max_{\pi} E\left[R_t | s_{t=s}, a_{t=a}, \pi\right]. \tag{3}$$

Then follows the Bellman optimality equation, we have the following:

$$Q^*(s, a) = E_{s' \sim S}\left[r + \gamma \max_{a'} Q(s', a') | s, a\right]. \tag{4}$$

The Q-value function is typically solved by iterating the Bellman equation in a traditional reinforcement learning framework as follows:

$$Q_{i+1}(s, a) = E_{s' \sim S}\left[r + \gamma \max_{a'} Q(s', a') | s, a\right]. \tag{5}$$

This formula automatically means when $i \longrightarrow \infty$, $Q_{i+1} \longrightarrow Q^*$,which suggests that if the state action value function is iterated continuously, it will ultimately converge on the optimal strategy:

$$\pi^* = \arg\max v^{\pi}(s), (\forall s). \tag{6}$$

In practice, however, iterating the Bellman equation to determine the Q-value is impractical in huge sample spaces due to the unimaginably large number of calculations.

2.2. Deep Reinforcement Learning Based on Value Function. Mnih was the first to propose a deep Q network (DQN) model by combining convolutional neural networks in deep learning and the Q learning algorithm in conventional reinforcement learning [18, 22]. This model incorporates a convolutional layer, which significantly improves the learning efficiency and performance [23].

Four preprocessed images preceding the current moment are fed into the DQN model. It becomes nonlinear after passing through three convolutional layers and two fully connected layers. Finally, the output layer produces the Q value associated with each action. TheDQN structure is depicted in Figure 2:

DQN made three significant improvements to the traditional Q learning algorithm to address the nonstationary as well as other issues associated with the nonlinear network used to represent the value function. DQN algorithm flow is shown in Figure 3.

DQN's training process makes use of the experience replay mechanism [24] (experience replay). The transferred samples $e_t = (s_t, a_t, r_t, s_{t+1})$. When an agent and environmental samples are transferred to a playback memory unit $D = \{e_1, \ldots, e_t\}$ at a given time t, they are stored there. Small batches of random transfer samples are selected from D each time, and the Stochastic Gradient Descent (the SGD) algorithm is used to update network parameter θ during training. When developing deep networks, it is common to require samples to be self-contained. Along with increasing the algorithm's stability, this random sampling method significantly reduces intersample relevance.

In this process, the agent and the transferred samples $e_t = (s_t, a_t, r_t, s_{t+1})$. obtained from environmental interaction are stored in the playback memory unit $D = \{e_1, \ldots, e_t\}$ at each time step t. When training deep networks, each sample must be completely independent of the rest of the data set. Random sampling improves the algorithm's stability by reducing intersample relevance.

In addition to the prior value function, DQN uses a deep convolutional network to estimate when another network is employed alone to obtain the desired Q value. The current

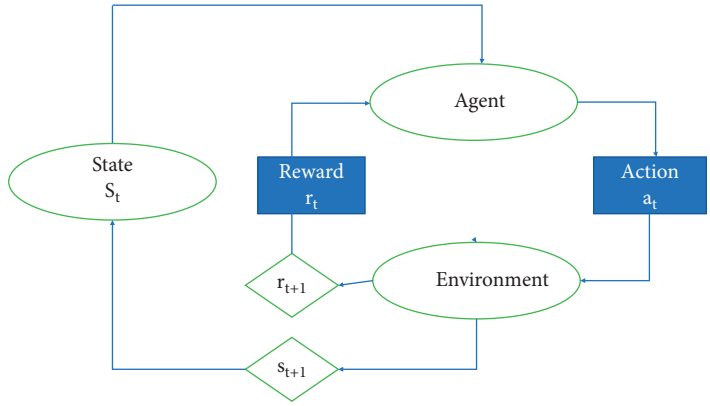

Figure 1: Flowchart of RL.

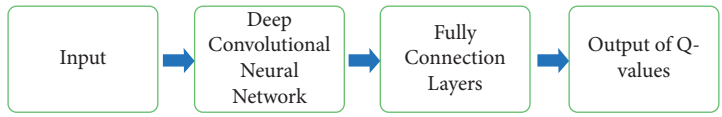

Figure 2: DQN structure.

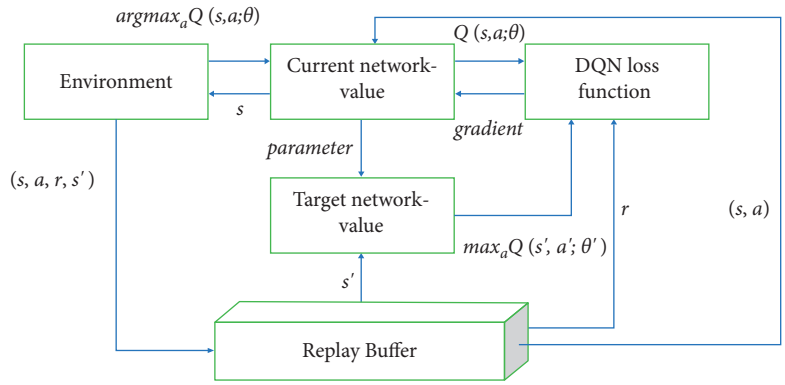

Figure 3: DQN algorithm flow.

value network's output, $Q(s, a|\theta_i)$, is used to calculate the value function of the current state action pair. The output of the target value network is represented as $Q(s, a|\theta_i^-)$ using the following formula:

$$Y_i = r + \gamma \max_{a'} Q(s', a'|\theta_i^-). \tag{7}$$

This is a rough representation of the value function optimization, with the purpose of obtaining the target Q value. The current value network's parameters θ are changed in real time. The parameters of the current value network will be used as the parameters of the target value network after N iterations. The principle of minimization reduces the joint variance of the desired Q value and the present Q value. The error function is as follows:

$$L(\theta_i) = E_{s,a,r,s'}\left[(Y_i - Q(s, a|\theta_i))^2\right]. \tag{8}$$

Differentiate the parameter θ to get the following gradient:

$$\nabla_{\theta_i} L(\theta_i) = E_{s,a,r,s^s}\left[(Y_i - Q(s, a|\theta_i))\nabla_{\theta_i} Q(s, a|\theta_i)\right]. \tag{9}$$

After introducing the target value network, the target Q value remains constant for a period of time, increasing the algorithm's stability by decreasing the correlation between the current and target Q values. By utilizing DQN, the reward and error terms are constrained to a small range, the Q and gradient values are guaranteed to be within a tolerable range, and the algorithm's stability is improved. DQN has been shown in experiments to be capable of resolving issues found in Atari 2600 games. When confronted with complex real-world situations [19], it demonstrates a competitive level comparable to that of human players. Even in less difficult nonstrategic games, DQN outperforms experienced human players. DQN uses the same network models, parameter settings, and training methods for visual perception as it does for auditory perception in the DRL task. This demonstrates how adaptable and flexible the DQN method can be.

3. Stock Prediction with RL

3.1. Deep Reinforcement Learning Based on the Policy Gradient for Stock Prediction. The policy gradient strategy (PG) is primarily achieved by modifying the settings in order to optimize the reward function. The key to this strategy is to alter the policy's parameters through extensive computation and iteration. Adjusted parameters will be included in the policy-making equation and will be modified on a continuous basis to accommodate more accurate policies. By fitting this approach, it will eventually converge on the optimal plan, which maximizes the reward value [15]. That is, in the optimal state, the strategy set contains specific parameters (or strategies) that ensure that the reward value of this strategy is equal to or greater than the reward value of any other strategy.

Parameterization is necessary to solve the DRL optimization problem. After constructing the fundamental equations in deep learning, the PG is typically used to pick parameters, alter the variable weights in the computer-constructed model, and finally choose the approach that maximizes the function's reward value. There are various advantages to this strategy, the primary one being that it reduces the computer's performance. This method streamlines the time-consuming dynamic iterative process, allows for a more precise optimization of the projected reward value, and also eliminates the need for heavy intermediate operations, which significantly reduces computing resource consumption. As a result, when compared to DQN and its advanced models, the DRL technique frequently achieves superior optimization results and has the lowest computing occupancy in model optimization when using the policy dimension reduction method. Additionally, the policy dimensionality reduction method may directly get the best policy from the policy set, significantly reducing computing costs. As a result, the bar for implementing this technology in practice is lower, and the area of its application is broader.

In practice, the strategy gradient method is a technique that uses an approximator directly to approximate and optimize the strategy, resulting in the optimal strategy. This method maximizes the strategy's projected total reward as follows:

$$\max_{\theta} E[R|\pi_{\theta}]. \tag{10}$$

$R = \sum_{t=0}^{T-1} r_t$ denotes the total number of rewards obtained in a plot. The most frequently used approach gradient is to enhance the likelihood of encountering a higher total reward plot. The strategy gradient approach operates in the following manner: We will assume here that the state, action sequence, and reward sequence of a complete plot are as follows, $\tau = (r_1, \ldots, s_{T-1}, a_{T-1}, r_{T-1}, s_0, a_0, r_0, s_1, a_1, s_T)$. The PG formula is then represented as follows:

$$g = R\nabla_{\theta} \sum_{t=0}^{T-1} \log\, \pi(a_t|s_t;\theta). \tag{11}$$

This gradient can be used to adjust policy parameters: $\theta \leftarrow \theta + \alpha g$. Among these, α determines how frequently policy parameters are updated, and it is called the learning rate. $\nabla_{\theta}\sum_{t=0}^{T-1}\log\pi(a_t|s_t;\theta)$, the gradient term, The gradient term denotes the potential direction in which the occurrence of a trajectory can be increased. The greater the total reward (in a single plot), the more "stretched" the probability density becomes after multiplying by R. The probability density will tend toward the trajectory with the highest total reward if a large number of paths with varying total rewards are collected, thereby increasing the likelihood of high-reward trajectories.

However, in some circumstances, the total reward R of each plot is positive, implying that all gradient g values are larger than or equal to 0. At this point, each trajectory τ met during the training process causes the probability density to "pull in" in a positive direction, significantly slowing the learning rate. This results in a relatively big variance for the gradient g. Thus, lower the variance of the gradient g by using any standardization operation in R. This strategy enables the algorithm to raise the likelihood of trajectories τ occurrence (for a greater total reward R) and decrease the probability of trajectories τ occurrence (for a lower total reward R).

Williams et al. [25, 26] proposed the REINFORCE method, which transformed the policy gradient into the following form:

$$\nabla_{\theta} \sum_{t=0}^{T-1} \log\pi(a_t|s_t;\theta)(R-b). \tag{12}$$

R's variance can be reduced by using an expected estimate of b as a baseline for the current trajectory. There is a strong correlation between R and b differences and the likelihood of a larger associated trajectory being chosen. For large-scale DRL challenges, the strategy parameters can be parameterized using deep neural networks and the optimal strategy can be solved using the standard strategy gradient method.

Additionally, another strategy to optimize strategies is to enhance the likelihood of performing "good" activities. In reinforcement learning, an advantage function is frequently used to quantify the quality of an action. As a result, the strategy gradient can be constructed using the following advantage function term:

$$g = \nabla_{\theta} \sum_{t=0}^{T-1} \widehat{A}_t \log\, \pi(a_t|s_t;\theta). \tag{13}$$

Among them, $\widehat{A}_t$ represents an estimate of (s_t, a_t) dominance function of the state action, which is usually constructed as follows:

$$\widehat{A}_t^{\gamma} = r_t + \gamma_{t+1} + \gamma^2 r_{t+2} + \cdots - V(s_t), \tag{14}$$

$\gamma \in (0,1)$ indicates discount factor. Sum of rewards with discount at this time $r_t + \gamma_{t+1} + \gamma^2 r_{t+2} + \cdots$ and the discounted state value function $V(s_t)$ is identical to R and the benchmark b in equation (12). When $\widehat{A}_t^{\gamma} > 0$, the likelihood of the appropriate action being chosen increases. When $\widehat{A}_t^{\gamma} < 0$, it will decrease the likelihood of the relevant action being chosen.

Figure 4: Training and testing results.

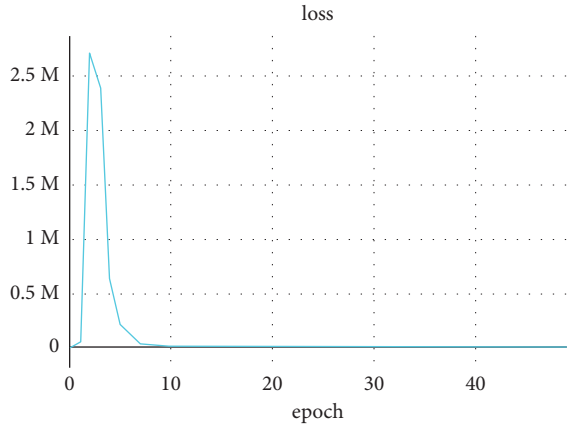

Figure 5: Loss function.

Figure 6: Reward convergence.

Hafner et al. [27] estimated the discounted reward summation using the value function, thereby lowering the gradient term's variance. At this point, the one-step truncated, $\widehat{A}_t^{\gamma}$'s expression is as follows:

$$\widehat{A}_t^{\gamma} = r_t + \gamma V(s_{t+1}) - V(s_t). \tag{15}$$

Additionally, the two-step truncated $\widehat{A}_t^{\gamma}$'s expression is as follows:

$$\widehat{A}_t^{\gamma} = r_t + \gamma_{t+1} + \gamma^2 V(s_{t+2}) - V(s_t). \tag{16}$$

However, this strategy introduces an element of estimating bias. To minimize variance while maintaining a little bias, Schulman et al. [28] proposed a generalized advantage function to solve this weakness as follows:

$$\widehat{A}_t^{y} = \delta_t + (\gamma\lambda)\delta_{t+1} + (\gamma\lambda)^2\delta_{t+2} + \cdots + (\gamma\lambda)^{T-t-1}\delta_{T-1}. \tag{17}$$

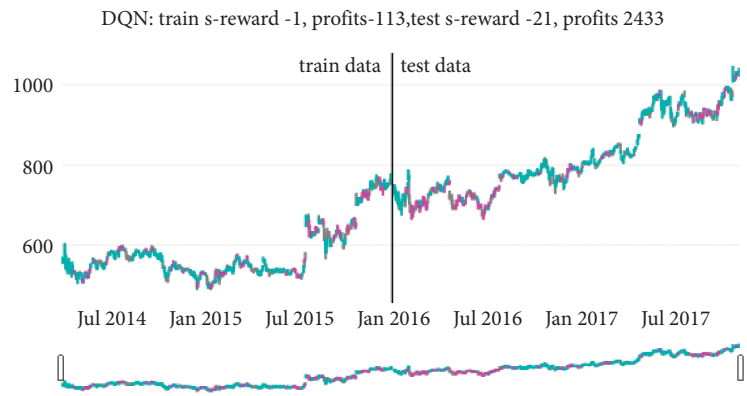

Figure 7: DQN training and testing results.

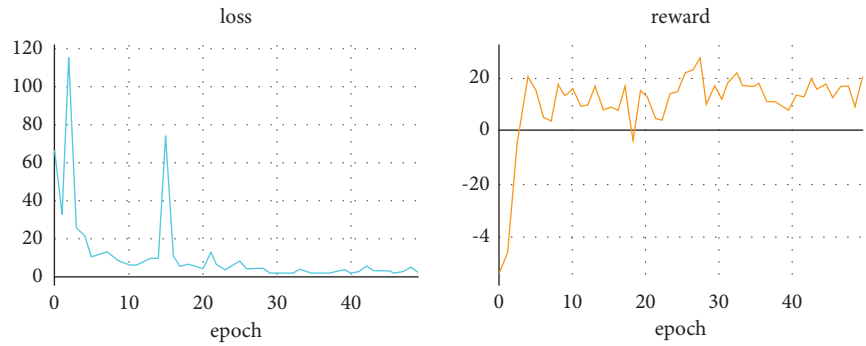

Figure 8: Loss function and convergence of DQN.

To take it a step further, Schulman et al. [29] then introduced a technique known as trust region policy optimization (TRPO). To broaden the application to large-scale state space DRL tasks, the TRPO method parameterizes the strategy using deep neural networks and achieves end-to-end control using only the original input image.

3.2. RL Application. In practice, during the day, when the trading system predicts that the stock price will continue to rise, there will be a judgement of whether to buy. When the stock price is predicted to fall continuously, a sell operation will be performed.

In the research, we use the single-agent training method. The rate of change of the price of a single stock in the market environment is what we focused, and we assumed that all the information in the market has been included in the environmental information and the state of the stock itself. Although the market is complex and changeable and the obtained market price is also the final expression of the game behavior of many Chinese investors. However, after a relatively long-term RL training, the result can theoretically reflect the information contained in the market more comprehensively, and therefore, this assumption is also harmless.

State is a unique parameter in the policy equation. Therefore, if it is assumed that the policies of all investors are inflexible, the choice of action is only based on a given state. Therefore, under 5-minute data, the policies of other investors can be regarded as the state of the market.

The indicators of stock information include the following: (1) Opening price, (2) Closing price, (3) Daily highest value, (4) Daily lowest value, (5) Daily average value, and (6) Trades Quantity. The above six data will serve as state indicators in the stock market environment.

s O: Opening price of the day.

s C: Closing price of the day.

s H: Highest price during the day.

s L: Lowest price during the day.

s A: Average price of the day.

Aq: the quantity and number of the successful trades of the day.

4. Results and Discussion

4.1. Simulated Results. The goal of this article is to establish a short-term stock price prediction framework with retrospective characteristics. In other words, the stock price fluctuations in the past are used as one of the references for predicting today's stock price. We used the historical daily prices and volumes of all U.S. stocks and ETFs to verify the methods we proposed. First, we trained the DRL model with the data from JUL 2014 to Jan 2016 and tested the model using the data from Jan 2016 to Jul 2017. The results are

shown in Figure 4As can be seen from the figure, in the training phase and the test phase, the output results of the model are very consistent with the actual stock prices.

In the process of using the data set to train the model, as the epoch data increases, the loss function will gradually decrease. During this training process, the training result of the loss function is shown in Figure 5. It can be seen from the figure that starting from 10 epoch data, the loss function tends to be the smallest.

In addition, in the process of training and testing the data set, we will observe the changes of reward in the DRL model, as shown in Figure 6. As illustrated in Figure 4, the reward changes dramatically at the start of training and testing, but stabilizes as the number of epochs increases, indicating that the DRL model is trained.

Following that, we predict the stock price using the DRL-based policy gradient method proposed in this paper, as illustrated in Figure 7. As illustrated in Figure 7, this paper's method is more accurate at forecasting the trend of stock price data. The results of analyzing the model's loss function and reward function are shown in Figure 8. When compared to the DRL results, it is discovered that while the method in this paper has poor loss function stability, it quickly stabilizes on the reward curve.

5. Conclusion and Discussion

To improve the accuracy of daily stock price predictions, this paper proposes a new reinforcement learning method that incorporates policy gradients. A comparison was made between the daily stock price changes predicted by the basic DRL and the daily stock price changes predicted by the proposed DRL based policy gradient method in this study. Despite the fact that the convergence time of the loss function is longer as a result of the use of different dimensionality reduction methods, the experimental results show that the accuracy of the new method in prediction as well as the stability of rapid prediction have both been significantly enhanced. This shows that our new method will be able to more easily capture the information of market changes. Therefore, it is more suitable for use in periods of turbulent market compared to traditional methods. However, this research still has room for improvement. For stock price research, the intraday volatility of stock prices is often large. Using the results of this research to trade can sometimes reduce the rate of return, which will cause certain losses to investors who hold a large number of stocks when they reduce their holdings. In addition, in terms of data volume, because there are only a few pieces of data generated every day, the data used to predict daily stock prices often have a long-time span, and market information in different years cannot be generalized. Therefore, raising the forecast frequency of daily stock prices to the 5-minute or even 1-minute level may be of greater reference for investors. Compared with the daily data, the data every five minutes have less information density, but they will more clearly reflect the overall picture of stock price fluctuations, which is conducive to the prediction of future stock price fluctuations and can also provide a better reference for investors to increase returns.

Data Availability

The datasets used and/or analyzed during the current study are available from the corresponding author on reasonable request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] F. Agostinelli, S. Mcaleer, A. Shmakov, and P. Baldi, "Solving the Rubik's cube with deep reinforcement learning and search," *Nature Machine Intelligence*, vol. 1, no. 8, pp. 356–363, 2019.

[2] S. Asadi, E. Hadavandi, F. Mehmanpazir, and M. M. Nakhostin, "Hybridization of evolutionary Levenberg-Marquardt neural networks and data pre-processing for stock market prediction," *Knowledge-Based Systems*, vol. 35, pp. 245–258, 2012.

[3] Y. Deng, Y. Kong, F. Bao, and Q. Dai, "Sparse coding-inspired optimal trading system for HFT industry," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 2, pp. 467–475, 2015.

[4] R. Hafner and M. Riedmiller, "Reinforcement learning in feedback control," *Machine Learning*, vol. 84, no. 1-2, pp. 137–169, 2011.

[5] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," *Advances in Neural Information Processing Systems*, vol. 12, 1999.

[6] G. Lample and D. S. Chaplot, "Playing FPS games with deep reinforcement learning," 2017, https://arxiv.org/abs/1404.3978.

[7] J. W. Lee, "Stock price Prediction Using Reinforcement Learning," in *Proceedings of the 2001 IEEE International Symposium on Industrial Electronics Proceedings*, pp. 690–695, Pusan, South Korea, June 2001.

[8] J. W. Lee, E. Hong, and J. Park, "A Q-Learning Based Approach to Design of Intelligent Stock Trading Agents," in *Proceedings of the 2004 IEEE International Engineering Management Conference*, pp. 1289–1292, Singapore, October 2004.

[9] J. Lehoczky and M. Schervish, "Overview and history of statistics for equity markets," *Annual Review of Statistics and Its Application*, vol. 5, no. 1, pp. 265–288, 2018.

[10] L.-J. Lin, *Reinforcement Learning for Robots Using Neural Networks (Tech. Rep.)*, DTIC Document, 1993.

[11] J. W. Lee, J. Park, J. O, J. Lee, and E. Hong, "A m approach to Q-Learning for daily stock trading," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 37, no. 6, pp. 864–877, 2007.

[12] B. Luo, D. Liu, H.-N. Wu, D. Wang, and F. L. Lewis, "Policy gradient adaptive dynamic programming for data-based optimal control," *IEEE Transactions on Cybernetics*, vol. 47, pp. 3341–3354, 2016.

[13] K. Miao, F. Chen, and Z. G. Zhao, "Stock price Forecast Based on Bacterial colony RBF Neural Network," *Journal of Qingdao University (Natural Science Edition)*, vol. 2, 2007.

[14] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Playing Atari with deep reinforcement learning," 2013, https://arxiv.org/abs/1312.5602.

[15] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[16] J. E. Moody, M. Saffell, Y. Liao, and L. Wu, *Reinforcement Learning for Trading Systems and Portfolios*, pp. 279–283, 1998, https://www.aaai.org/Papers/KDD/1998/KDD98-049.pdf.

[17] M. Nabipour, P. Nayyeri, H. Jabani, A. Mosavi, E. Salwana, and S. Shahab, "Deep learning for stock market prediction," *Entropy*, vol. 22, no. 8, p. 840, 2020.

[18] M. P. Naeini, H. Taremian, and H. B. Hashemi, "Stock market value prediction using neural networks," in *Proceedings of the 2010 International Conference on Computer Information Systems and Industrial Management Applications*, pp. 132–136, IEEE, Krakow, Poland, October 2010.

[19] E. S. Olivas, J. D. M. Guerrero, M. Martinez-Sober, J. R. Magdalena-Benedito, and L. Serrano, *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques: Algorithms, Methods, and Techniques*, IGI global, Hershey, PA, USA, 2009.

[20] D. S. S. Pinto and K. R. G. Da Silva, "Robot position control in pipes using *Q* learning," in *Proceedings of the 2016 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 004609–004613, IEEE, Budapest, Hungary, October 2016.

[21] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proceedings of the 32nd International Conference on Machine Learning*, pp. 1889–1897, PMLR, Lille, France, July 2015.

[22] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," 2015, https://arxiv.org/abs/1506.02438.

[23] R. S. Sutton, D. Mcallester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in Neural Information Processing Systems*, vol. 12, 1999.

[24] H. Wang, Y. Wu, G. Min, J. Xu, and P. Tang, "Data-driven dynamic resource scheduling for network slicing: a deep reinforcement learning approach," *Information Sciences*, vol. 498, pp. 106–116, 2019.

[25] C. J. C. H. Watkins, "Learning from delayed rewards," 1989.

[26] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[27] P. Wolf, C. Hubschneider, M. Weber et al., "Learning How to Drive in a Real World Simulation with Deep Q-Networks," in *Proceedings of the 2017 IEEE Intelligent Vehicles Symposium*, pp. 244–250, IEEE, Los Angeles, CA, USA, July 2017.

[28] T. Xu, S. Zou, and Y. Liang, "Two time-scale off-policy TD learning: non-asymptotic analysis over Markovian samples," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[29] Y. Yuan, Z. L. Yu, Z. Gu et al., "A novel multi-step Q-learning method to improve data efficiency for deep reinforcement learning," *Knowledge-Based Systems*, vol. 175, pp. 107–117, 2019.