IJCNN 2019. International Joint Conference on Neural Networks. Budapest, Hungary. 14-19 July 2019

# Quantitative Trading on Stock Market Based on Deep Reinforcement Learning

Jia WU, Chen WANG, Lidong XIONG, Hongyong SUN
*School of Information and Software Engineering*
*University of Electronic Science and Technology of China*
ChengDu, China
jiawu@uestc.edu.cn; wc00533@gmail.com; xionglidong_2019@outlook.com; sunhongyong0@gmail.com

***Abstract*—With the development of computer science technology and artificial intelligence, quantitative trading attracts more investors due to its efficiency and stable performance. In this paper, we explore the potential of deep reinforcement learning in quantitative trading. A LSTM-based agent is proposed to learn the temporal pattern in data and automatically trades according to the current market condition and the historical data. The input to the agent is the raw financial data and the output of the agent is decision of trading. The goal of the agent is to maximize the ultimate profit. Besides, to reduce the influence of noise in the market and to improve the performance of the agent, we use several technical indicators as an extra input. The proposed system has been back-tested on the stock market. The results demonstrate that our method performs well in most conditions.**

***Index Terms*—Quantitative trading, Deep learning, Reinforcement learning, Deep reinforcement learning**

## I. Introduction

Chinese security market, which is one of the world's major capital markets, has developed more than 20 years since the 1980s. Recently, quantitative trading has attracted more and more investors. Quantitative trading employs mathematical models and artificial intelligent models to analyze the financial data without necessarily establishing a link to financial theory. Based on these models, it issues trading orders by a computer programs, in order to obtain maximum profits. Compared with traditional strategies, quantitative trading possesses much more advantages: quantitative trading systems can timely track changes in the market and respond quickly. The trading strategies are designed based on mathematical models learned from long-term historical data, not rules defined by human. Hence, the strategies are more reliable. Quantitative trading can overcome negative emotions of human traders, such as greed, fear, etc.

There are a bunch of quantitative trading methods. Many of them focus on modeling the dynamics of the market [1]–[3] and make decisions based on these models. However, since the complexity and the changing behavior of the market participants, it is hard for simple models to capture all important properties of the market. Hence, the decisions based on models are prone to failures.

Another branch of researches concentrates on treating the quantitative trading as decision-making problem which trains an electronic agent to directly make decisions. The challenges presented by this method are financial signal representations and optimal action execution.

The first one is how to summarize the stock market state. The stock data is often represented as the highly non-stationary time series that contain a multitude of noise, jump, and movement. In order to reduce data noise and uncertainty, manual financial characteristics such as moving average or stochastic technical indicators are used to summarize the stock market state [4]. The research for superior indicators for technical analysis [5] has been widely discussed in quantitative trading. However, these indicators exhibit relatively poor generalization. For example, the moving average indicator is sufficient to describe the trend, but may suffer significant losses in the mean-reversion market [6]. These limitations cry out for a robust data-driven approach, which learn features directly from the financial data instead of predefined manual features designed by human. The second challenge is owing to the dynamic behavior of trading decision. Executing trading orders is a systematic task that requires amount of practical factors to be considered. Therefore, in addition to the current market condition, the policy learning component also requires a specific modeling of the historical behavior and the corresponding trading positions.

Reinforcement learning (RL) is an effective framework for solving decision-making problem. RL is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize the cumulative reward. RL has been successful applied in several domains, including robot navigation [7], helicopter control [8], etc. During the past years, there have been several methods that apply RL to solve the quantitative trading problem. [9], [10] make use of a Q-learning approach. [11], [12] apply the method to derivatives pricing applications. [13] employs a TD($\lambda$) for evaluating actions. Moody [14] proposes recurrent reinforcement learning for discovering investment policies. However, the function of agent in traditional RL-based methods is too simple to capture the dynamics of the market. Besides, the agent is not flexible enough for exploring policies.

Deep learning is lately drawing much attention due to its ability to search the hidden patterns in high dimensional data [15]. Combined with RL, deep reinforcement learning (DRL) has a great ability to learn control policies directly

This work was supported by The National Natural Science Foundation of China (grant 61503059).

from the high dimensional data. In this paper, we present a quantitative stock trading algorithm based on DRL, which is mainly composed of deep learning and reinforcement learning, which aim to learn stock feature and to learn optimal strategy, respectively. Although deep learning has shown excellent performance in many signal processing problems such as image and speech recognition, as far as we know, the combination of deep learning and reinforcement learning in the field of financial quantitative trading still needs to be further studied.

In our method, the trading agent is constructed by LSTM recurrent neural network. The reason we use this network is that large LSTM are highly expressive models that can learn rich temporal representation of data. The LSTM based agent can explore the temporal pattern in the stock data and remember the historical behaviors. In the part of experiment, we have compared the performance of LSTM-based agent and the fully connected neural network based agent. Besides, we have compared the performances of different combinations of features and select the best combination for the trade. After training the agent, it is backtested on real dataset. The backtesting results verify that the LSTM based agent performs much more stable in the changing dynamics of market conditions and could obtain stable profits in most conditions.

The remaining part of this paper is organized as follows: Section II introduces the theory foundation of deep reinforcement learning and reviews some related works. Section III describes the structure of the agent and how we train it. The performance of the proposed model will be verified in section IV. In this part, we will also study the impact of different combinations of indicators. Section V concludes this paper and discuss directions of future research.

## II. RELATED WORKS

### A. Deep Learning

Artificial neural network (ANN) is a computing systems composed of a collection of connected non-linear units to process data. Traditional ANNs are generally restricted to only a few layers that limits their ability to express complex relationship. Deep learning are ANN which has relatively high depth and is much highly expressive. However, it faces several challenges when applying back-propagation and gradient descent to train models. In 2006, Hinton [16] proposed a pre-training method to solve the problem [17] . After that, various deep learning models have emerged, such as Stacked Auto-Encoder [18] , Restricted Boltzmann Machine [19] , Deep Belief Network [20] and Recurrent Neural Network [21] , and so on.

Recurrent neural network is a kind of ANN where connections between units form a directed cycle. This architecture creates an internal state of the network which allows it to process time series data and remember the temporal relation. However, it cannot solve the long-term dependence problem. Hochreiter and Schmidhuber proposed Long Short-Term Memory Neural Network (LSTM) [22] to avoid this problem. The LSTM network is a special recurrent neural network that maintains the information at arbitrary time by three gates: input gate, forget gate, and output gate.

### B. Reinforcement Learning

Reinforcement learning is a computational approach for understanding and automating goal-directed learning and decision-making [23]–[26]. RL defines a framework where a learning agent interacts with its environment in terms of states, actions and rewards. At each time, the agent observes the state of the environment and executes an action from a set of legal actions. The execution of action is passed to the environment and modifies the internal state of the environment. The environment emits a reward to reflect how good the action is. The goal of the agent is to maximize the cumulative reward.

Algorithms of RL are generally categorized into two types, model-based and model-free. Model-based RL models the dynamics of the environment and makes decision based on predicting the following states and rewards. The drawback of model-based RL is that a model of the environment is fundamentally hard to learn. Model-free algorithms don't depend on a model of the environment. They are more popular since they are easier to implement and train. There are two main approaches to train agents with model-free RL, policy optimization and Q-learning. Q-learning [27] methods try to approximate the Q value which qualifies the action and the state pair. The optimal control policy is learned from the Q value. The policy optimization methods learn the control policy directly.

In this paper, we use policy optimization method to train the trader agent. Since compared to Q-learning methods, it is much simpler that only requires a differentiable objective function with latent parameters. In addition, rather than describing diverse market conditions with some discrete states (in Q-learning), the policy optimization method can learn the policy directly from the continuous sensory data (market characteristics). In short, the policy optimization method exhibits two advantages: flexible objective for optimization and continuous descriptions of market condition. So it is a better framework for trading than the Q-learning approaches.

### C. Deep Reinforcement Learning

As the development of deep learning, Googles AI team integrates deep learning with Q-learning, namely Deep Reinforcement Learning (DRL) [28], which can solve complex tasks such as playing Atari games. DRL enables end-to-end learning from perception to action, where deep neural network is used to represent value function (Q-Learning), policy (policy optimization) or the environment model. Since the high expressive ability of deep neural network, DRL agent can learn the representation from the high-dimensional data and solve more complex decision problem. To the best of our knowledge, there is not too much work concerned the quantitative trading based on DRL. This paper aims to generalize the power of deep learning into a new field for financial signal processing and learning. We will apply DRL to design a trading system for stock market.

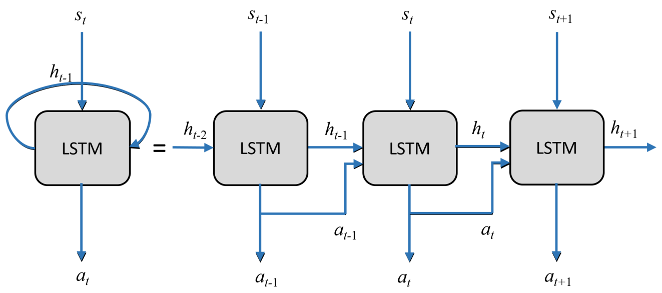


Fig. 1. Model structure of the trading agent ($s_t$ is the state of market; $h_t$ is the hidden state of LSTM; $a_t$ is the action taken at time $t$; the input to LSTM is $s_t$ concatenated with $h_t$ and $a_t$)

## III. DRL FOR QUANTATITIVE TRADING

In this section, we will describe how to apply DRL to realize quantitative trading. First, we will show the model structure of the trading agent. Then, we will explain how to design the state, the action set and the reward function. Finally, we will illustrate the training process of the agent.

### *A. Model of Trading Agent*

The trading agent works as follows: at each time, it obtains the current market state and integrates the historical data to predict the future state. Based on all information, it makes the real-time trading decision. The immediate profit is obtained based on the decision, which is used to update the internal parameter of the trading agent. After multiple iterations, the agent learns how to improve its decisions.

According to the trading process, the trading agent should have two functional parts: a memory component that makes predictions future state based on historical information, and a decision-making component that decides what actions to take. In the paper, the trading agent is constructed by a LSTM network. The reason we use only one deep neural network to implement the two functions are:

- The input to the agent should include not only the observation from the market at each time frame, but also the information about what happens over time. LSTM network can remember the interrelation between time series data.
- Search space can be greatly reduced. If the search space is large, the multi-layer dense network may be too big to train; while the structure and parameters of the LSTM network are shared at any time, this greatly reduces the difficulty of training.

The structure of the trading agent is illustrated in Fig.1. The input to the agent includes the current state of market $s_t$ and the hidden state $h_{t-1}$ of LSTM, which remembers the temporal relation of historical data. Besides, the decision $a_{t-1}$ at previous time is also fed into LSTM as an extra input. The output of the agent is the decision $a_t$ at each time.

### *B. State, Action and Reward Design*

In this paper, we consider agent trading fixed position size in a single security. The method can be generalized to the trading with varying quantities.

The state of the agent includes the volume and the price series being traded. Since the market is stochastic and the price series contains a large amount of noise, the state also includes technical indicators which summarizes the trend or the dynamics of the market, as shown in Table II. It is noted that the trend presented by technical indicators are sometimes contradictory. Hence, the most useful combination of indicators has to be finely selected. In the experiment part, we will test different combinations of indicators to find the best one.

Our trading agent is assumed to take long, neutral, and short positions, $a_t \in \{long, neutral, short\} = \{1, 0, -1\}$. We assume that the outcome of a neutral position will have an effect on the traders profit.

The goal of the agent is to maximize the ultimate profit. We assume that the agent trades at the end of each time interval $t$. A reward which evaluates the profit at the time $t$, is defined as:

$$r_t = \begin{cases} \Delta n \times (p_t^c - p_t^o) + c_t, & a = 1 \\ \Delta n \times (p_t^o - p_t^c) + c_t, & a = -1 \\ N \times p_t^c + M - I, & a = 0 \end{cases} \qquad (1)$$

Where $p_t^o$ and $p_t^c$ represent the opening price and closing price of the day $t$, respectively. $\Delta n$ is the change volume of trading; $N$ is the current amount of stock that the agent holds. $M$ is the current capital and $I$ is the initial capital. $c_t$ denotes the transaction cost incurred at time $t$.

### *C. Training Algorithm*

A policy $\pi$ is a rule used by the agent to decide what actions to take. Suppose $\theta$ represent the parameter of the policy $\pi_\theta$. A trajectory $\tau$ is a sequence of states, actions and rewards: $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \dots)$. Return is the cumulative reward over a trajectory $R(\tau) = \sum_{t=0}^{\infty} r_t$. The goal of the agent is to maximize the expected return:

$$\max_\theta J(\theta) = \max_\theta E_{\tau\sim\pi_\theta}[R(\tau)] \qquad (2)$$

According to the policy gradient algorithm [23], $\nabla_\theta(J(\theta))$ is calculated by:

$$\begin{aligned} & \nabla_\theta(J(\theta)) \\ = \ & \nabla_\theta E_{\tau\sim\pi_\theta}[R(\tau)] \\ = \ & \nabla_\theta \textstyle\sum_k [P(\tau_k|\theta)R_k] \\ = \ & \textstyle\sum_k \left[P(\tau_k|\theta)\dfrac{\nabla_\theta P(\tau_k|\theta)}{P(\tau_k|\theta)}R_k\right] \\ = \ & \textstyle\sum_k [P(\tau_k|\theta)\nabla_\theta log P(\tau_k|\theta)R_k] \\ = \ & E_{P(\tau|\theta)}[\nabla_\theta log P(\tau|\theta)R] \\ = \ & E_{P(\tau|\theta)}\left[\textstyle\sum_{t=1}^{\infty}\nabla_\theta log P(a_t|a_{(t-1)})R\right] \end{aligned} \qquad (3)$$

Where $R_k$ represents the reward of the $k^{th}$ trajectory $\tau_k$; $P(\tau|\theta)$ denotes the probability of generating a trajectory $\tau$.

Equation (3) is an unbiased estimator of gradient. It can be estimated by sampling $m$ times. To reduce variance of this estimation, we use a baseline function $b$:

$$\nabla_\theta(J(\theta)) \approx \frac{1}{m}\sum_{k=1}^{m}\sum_{t=1}^{T}\nabla_\theta logP(a_t|a_{(t-1)})(R_k - b) \quad (4)$$

Where $b$ is an exponential moving average of the return of previous trajectories.

## IV. EXPERIMENTAL RESULTS

- **Dataset** The stock data utilized in this experiment is obtained through the Tushare interface [29], a free and open source financial data package. It realizes the process of data collecting, data cleaning and data storage, greatly reduces the work in data preprocessing and provides the data analysts with clean data to facilitate the analysis and building models. We chose the Tushare interface to obtain the stock data based on the following three considerations:
  1) Tushare interface package is a widely used package with high credibility and high quality in the field of financial data analysis.
  2) The data formats returned by the Tushare interface is very convenient for data analysis and visualization.
  3) Tushare has rich amount of fanatical data, such as stocks, funds, futures, crypto-currencies, etc.

  Our trading system captures the day-level stock data in Chinese stock market, the closing and the opening prices, the highest and the lowest prices. We have selected six stock data from 2015-10-25 to 2017-10-25 (stock codes 002415, 600016, 600028, 600547, 600999 and 601988), which exhibit quite different market trends, as shown in Fig 2. Fig.2 shows the original financial data contains mainly three trends.
- **Network structure and training details** The LSTM-based agent is composed of an input layer, 5-layer LSTM with 31 hidden units on each layer, a dense layer and a soft-max layer. The update rate of the baseline function $b$ is 0.8. Weights of the agent are initialized uniformly between -0.2 and 0.2. The neural network of the agent is trained with Adam optimizer [30] with a learning rate of 0.001.

### A. Comparison between LSTM-based agent and fully connected neural network based agent

In this part, we will compare the performance of the LSTM-based agent and the fully connected neural network based agent (FC-based agent). The fully connected neural network consists of four layers: an input layer; a fully connected layer with 32 nodes; a dropout layer with 64 nodes; a soft-max layer with three output unites. The learning rate is 0.007 with a decay rate 0.99. The Adam Optimizer is used to optimize the loss function.

The data input to the two agents includes the opening and closing prices at each day, the volume of historical trending data. The initial capital is 500,000 China Yuan (CNY), and the position size of trading at each time is fixed to 10,000 CNY. The performance is evaluated by the final profit after 10,000 episodes of training. Fig.3 presents the performance of two agents during the training phase. Table I summarizes the final profit that each agent can achieve after training.

TABLE I
COMPARISONS OF PERFORMANCE ON FC AND LSTM

| Stock code | Ultimate profit (CNY) | | Average profit (CNY) | |
|---|---|---|---|---|
| | LSTM | FC | LSTM | FC |
| 600547 | 725, 845 | 570, 109 | 845, 080 | 578, 903 |
| 600028 | 541, 078 | 492, 508 | 569, 120 | 500, 547 |
| 600999 | 820, 800 | 767, 521 | 801, 900 | 718, 053 |
| 601988 | 730, 506 | 691, 030 | 757, 934 | 678, 066 |
| 002415 | 1, 900, 132 | 1, 921, 433 | 1, 820, 798 | 1, 703, 645 |
| 600016 | 1, 310, 378 | 1, 315, 043 | 1, 153, 643 | 1, 159, 678 |

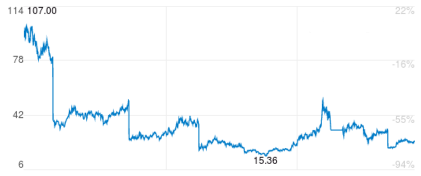
We can see that both agents can achieve much more profits in the market since the high expressive ability of deep neural network. If the stock data has a large fluctuation of movements, the fully connected neural network performs poor. On the contrary, the LSTM-based agent can perform much better. The reason is that the LSTM-based agent can remember the temporal relation of data. The experiment results demonstrate that LSTM is suitable for constructing a trading agent.

### B. Comparisons of different combinations of indicators

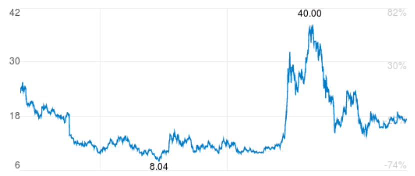
Since the prices of the market contain too much noise, we will add several technical indicators to capture the main trend. In technical analysis, a technical indicator is a mathematical calculation based on historic price, volume, or (in the case of futures contracts) open interest information that aims to forecast financial market direction. Many technical indicators have been developed and new variants continue to be developed by traders with the aim of getting better results. We consider the combinations of the indicators [6] presented in Table II.

It is noted that there is no indicator that can work well in all conditions. Indicators can present a significant trend in certain conditions and in other conditions, they point a wrong trend. Hence, we have to carefully select a combination of indicators. In the following experiment, we will test four combinations of indicators, as shown in Table III. In the table, nopen, nclose, nhigh, nlow represent the opening price, the closing price, the maximum price and the minimum price of a stock at each day, respectively. And ivolume represents the volume of the trading stock. The input to the LSTM-based agent contains the daily opening and closing , the highest and the lowest prices, the volume of trading, and a combination of indicators. The performance during training is presented in Fig.4. Table IV shows the ultimate profit and the average profit after training.

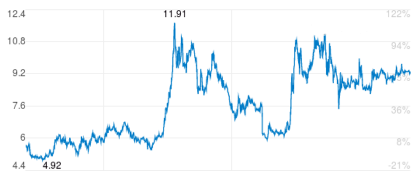
From the experiment results, we can see that different combinations of technical indicators have a greater impact on the performance of the agent and the ultimate profit. The trading agent with d2 generally outperforms other combinations of indicators, followed by d1, d3 and d4. The combination d1 has only 10 indicators, which results in a large variance of reward.



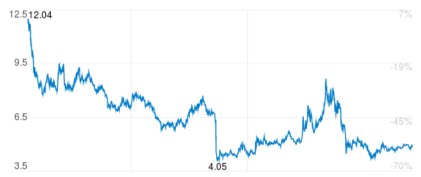
(a) Stock code 002415 shows a large downward movement



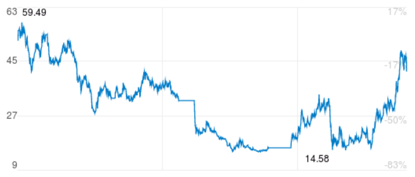
(b) Stock code 600999 shows an over 100% upward movement



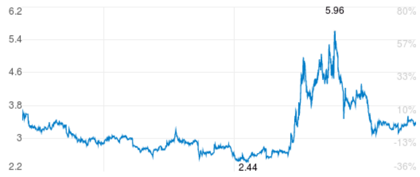
(c) Stock code 600016 has a fluctuated movement



(d) Stock code 600028 has a fluctuated movement

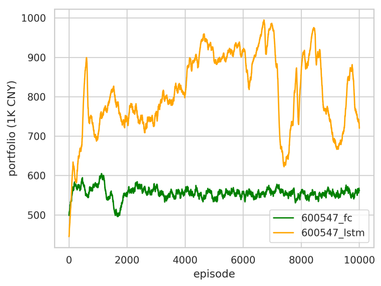


(e) Stock code 600547 has a slow downward trend and goes upward after reaching the bottom

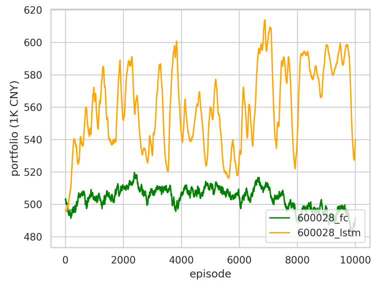


(f) Stock code 601988 has a large upward movement

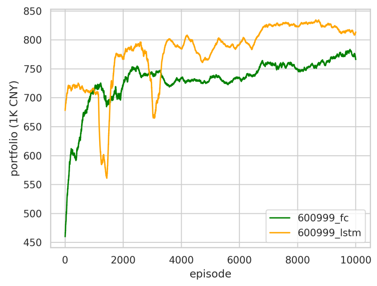
Fig. 2. Prices based on day resolutions of six stock data during 2005-2017 contains different trends



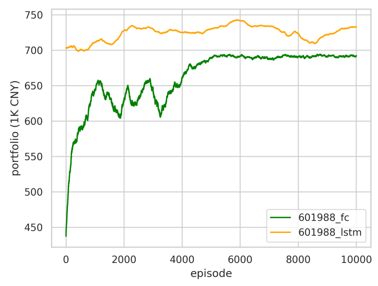
(a) Stock code 600547



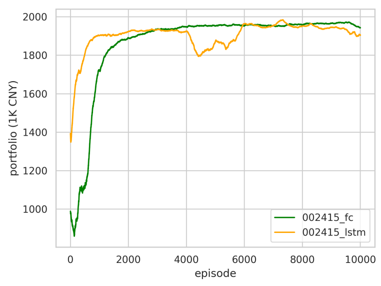
(b) Stock code 600028



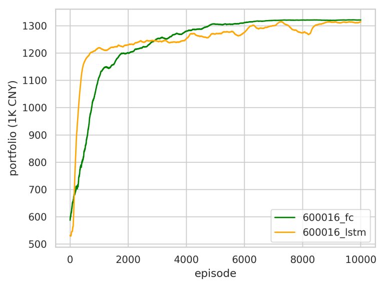
(c) Stock code 600999



(d) Stock code 601988



(e) Stock code 002415



(f) Stock code 600016

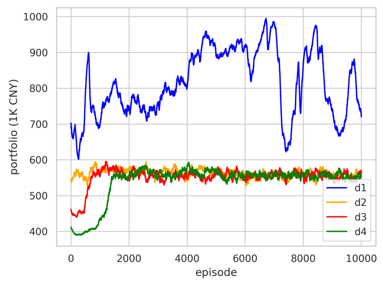
Fig. 3. Comparing the cumulative reward between LSTM-based agent and FC-based agent on different stocks

TABLE II
FINANCIAL TECHNICAL INDICATORS

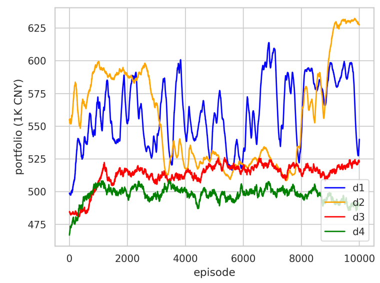
| Indicators | Full name | Indicators | Full name |
|---|---|---|---|
| DEA | Data Envelopment Analysis | MACD | Moving Average Convergence/Divergence |
| EXPMA | Exponential Moving Average | CDP | GDP deflator |
| TRIX | Triple Exponentially Smoothed Average | BBI | Bull And Bear Index |
| ASI | Accumulation Swing Index | KDJ | Stochastics |
| RSI | Relative Strength Index | PSY | Psychological Line |
| VR | Volatility Volume Ratio | ADX | Average Directional Index |
| CCI | Commodity Channel Index | WR | Williams Overbought/Oversold Index |
| dm_up | Directional movement up | dm_down | Directional movement down |

TABLE III
COMBINATIONS OF TECHNICAL INDICATORS

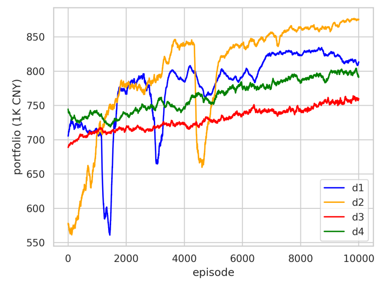
| Combination of indicators | Quantity | Indicators |
|---|---|---|
| d1 | 10 | nclose, nopen, nhigh, nlow, K, D, J, RSI, WR6, WR12 |
| d2 | 31 | nclose, nopen, nhigh, nlow, DEA, MACD_bars, Upper_Band, Lower_Band, EMA26, CDP, BIAS6, BIAS12, BIAS24, EXPMA5, EXPMA10, EXPAM20, TRIX, BBI, ASI, K, D, J, RSI, PSY, VR, dm_down, dm_up, ADX, ADRX, WR6, WR12 |
| d3 | 34 | nclose, nopen, nhigh, nlow, ivolume, DEA, MACD_bars, Upper_Band, Lower_Band, EMA26, CDP, BIAS6, BIAS12, BIAS24, EXPMA5, EXPMA10, EXPAM20, TRIX, BBI, ASI, K, D, J, RSI, PSY, VR, dm_down, dm_up, ADX, ADRX, WR6, WR12, CCI, AR |
| d4 | 35 | nclose, nopen, nhigh, nlow, ivolume, DEA, MACD_bars, Upper_Band, Lower_Band, EMA26, CDP, BIAS6, BIAS12, BIAS24, EXPMA5, EXPMA10, EXPAM20, TRIX, BBI, ASI, K, D, J, RSI, PSY, VR, dm_down, dm_up, ADX, ADRX, WR6, WR12, CCI, AR, BR |



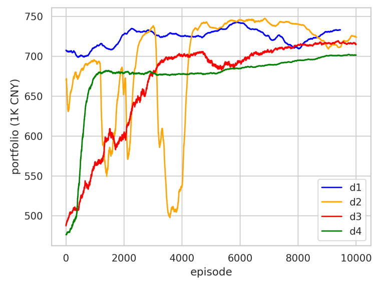
(a) Stock code 600547



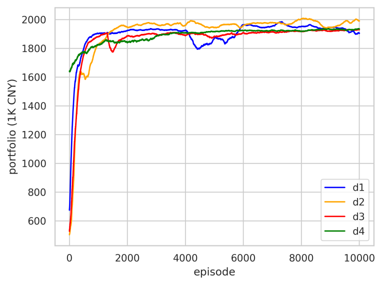
(b) Stock code 600028



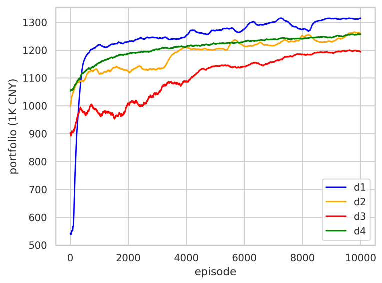
(c) Stock code 600999



(d) Stock code 601988



(e) Stock code 002415

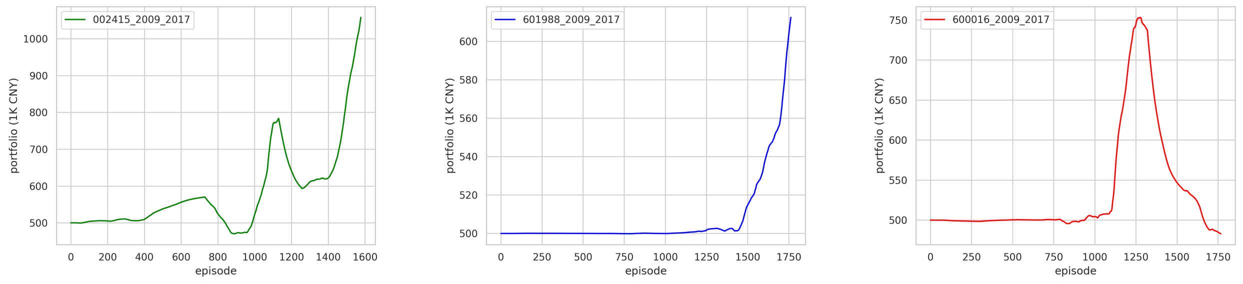


(f) Stock code 600016

Fig. 4. Comparing the profits of different combinations of indicators on LSTM-based agent

TABLE IV
COMPARISONS OF DIFFERENT COMBINATION OF INDICATORS

| Stock code | Ultimate profit (CNY) | | | | Average profit (CNY) | | | |
|---|---|---|---|---|---|---|---|---|
| | d1 | d2 | d3 | d4 | d1 | d2 | d3 | d4 |
| 600547 | 723596 | 563497 | 575342 | 568342 | 820346 | 587495 | 574632 | 569021 |
| 600028 | 538023 | 627092 | 524780 | 488326 | 580153 | 572064 | 513603 | 499821 |
| 600999 | 810772 | 875334 | 762005 | 793490 | 720868 | 796602 | 726121 | 764402 |
| 601988 | 730245 | 725969 | 714056 | 701043 | 720056 | 729968 | 654390 | 698900 |
| 002415 | 1905245 | 1999983 | 1935892 | 1937123 | 1653269 | 1980020 | 1648975 | 1825547 |
| 600016 | 1310085 | 1254377 | 1199765 | 1252932 | 1143932 | 1178946 | 1053670 | 1187643 |



(a) Stock code 002415 (b) Stock code 601988 (c) Stock code 600016

Fig. 5. Backtesing results of agent on three stocks from 2009 to 2017

## *C. Backtesting*

After training the agent, we backtest the model. A total 13 years of day-level data from October 25, 2005 to October 25, 2017 are collected for back testing the on-line performance of our agent. The first 4 years of data are used to set up the parameters of the agent. The test period is the rest 9 years from October 25, 2009 to October 25, 2017. For each year during 2009 through 2017, the sliding window of the training data covers two years and moves forward one year covering a new training set. Within the sliding window, the previously learned parameters are used to guide the trading for the upcoming year. Therefore, the agent can get aware of the changing stock market. The back testing results from 2009 through 2017 are shown in Fig. 5. From the results, we can see that our agent could make profit in most cases. However, it is observed that the profit of the stock code 600016 decreases sharply at the end since the price of that stock has a fluctuated movement. Such performance degradation implies that the difficulty of precisely capturing the dynamics of the market.

## V. Conclusion

This paper combines deep learning with reinforcement learning for stock signal representation and online trading. In this framework, LSTM-based agent can automatically sense the dynamics of the stock market and alleviate the difficulty of manually designing indicators from massive data. Moreover, we have added financial indicators into the LSTM network to reduce the influence of the market noise. Reinforcement learning algorithm is used train the agent to learn trading policy. The experiment results verify the potential of our method. There are some research directions to be concentrated on in the future. Although we have added some technical indicators to reduce the influence of the noise in the stock market, there is much inherent uncertainty in the stock data. How to reduce the uncertainty is still challenging. In addition, how to make use of the experiences from the past training data is still needed to be further studied.

## References

[1] J. B. Heaton, N. G. Polson, and Jan Hendrik Witte. “Deep learning for finance: deep portfolios”. Applied Stochastic Models in Business and Industry, ISSN 1526-4025, 2016.

[2] S. Niaki and S. Hoseinzade. “Forecasting S&P 500 index using artificial neural networks and design of experiments”. Journal of Industrial Engineering International, ISSN 2251-712X, 2013.

[3] Fabio D Freitas, Alberto F De Souza, and Ailson R de Almeida. “Prediction-based portfolio optimization model using neural networks”. Neurocomputing, vol.72, no.10, pp.2155C2170, 2009.

[4] C. J. Neely, D. E. Rapach, J. Tu, and G. Zhou, “Forecasting the equity risk premium: The role of technical indicators”, Manage. Sci., vol.60, no.7, pp.1772-1791, 2014

[5] J. J. Murphy. Technical Analysis of the Financial Markets: “A Comprehensive Guide to Trading Methods and Applications”. New York, NY, USA: New York Institute of Finance, 1999.

[6] J. M. Poterba and L. H. Summers, Mean reversion in stock prices: Evidence and implications, J. Financial Econ., vol. 22, no. 1, pp. 27-59, 1988.

[7] H. R. Beom and K.S.Cho, A sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning, IEEE Trans. Syst., Man, Cybern., vol.25, no.3, pp.464-477, Mar.1995.

[8] H.J. Kim, M. I. Jordan, S.Sastry, and A.Y.Ng, Autonomous helicopter flight via reinforcement learning, in Proc. Adv. Neural inf. Process. Syst., pp.799-806, 2003. [

[9] R. Neuneier, Optimal asset allocation using adaptive dynamic programming, in Advances in Neural Information Processing Systems, D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, Eds. Cambridge, MA: MIT Press, vol. 8, pp. 952C958, 1996,

[10] R. Neuneier and O. Mihatsch, Risk sensitive reinforcement learning, in Advances in Neural Information Processing Systems, M. S. Kearns, S. A. Solla, and D. A. Cohn, Eds. Cambridge, MA: MIT Press, vol. 11, pp. 1031C1037, 1999.

[11] Roy, John N. Tsitsiklis and Benjamin Van, Optimal stopping of Markov processes: Hilbert space theory, approximation algorithms, and an application to pricing high-dimensional financial derivatives, IEEE Trans. Automat. Contr., vol. 44, no.10, pp. 1840C1851, Oct. 1999.

[12] Tsitsiklis J N , Van R B, Regression methods for pricing complex American-Style Options, IEEE Trans. Neural Networks, vol. 12, no. 4, pp. 694C703, July 2001.

[13] B. Van Roy, Temporal-difference learning and applications in finance, in Computational Finance 1999, Y. S. Abu-Mostafa, B. LeBaron, A. W. Lo, and A. S. Weigend, Eds. Cambridge, MA: MIT Press, pp. 447C461, 2001,

[14] J. Moody, L. Wu, Y. Liao, and M. Saffell, Performance functions and reinforcement learning for trading systems and portfolios, J. Forecasting, vol. 17, pp. 441C470, 1998.

[15] Yann LeCun, Yoshua Bengio and Geoffrey Hinton, Deep Learning, Nature, vol. 521, pp. 436C444, 28 May 2015.

[16] Hinton G E, Osindero S, Teh Y W. A fast learning algorithm for deep belief nets. Neural Computation, vol.18, no.7, pp.1527-1554, 2006.

[17] Q. Liu, J.W. Zhai, Z.C.Zhang et.al, A survey on deep reinforcement learning. Chinese Journal of Computers, vol.40, no.1, pp.3, 2018.

[18] Vincent P, Larochelle H, Lajoie I, et al. Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. Journal of Machine Learning Research, 2010, 11(6): 3371-3408.

[19] Sutskever I, Hinton G E, Taylor G W. The recurrent temporal restricted boltzmann machine. Proceedings of the 27th Annual Conference on

Neural Information Processing Systems. Vancouver,vol.09, pp.1601-1608, Canada.

[20] Lee H, Grosse R, Ranganath R, et al. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. Proceedings of the 26th International Conference on Machine Learning. Montreal, Canada, pp.609-616, 2009.

[21] Mikolov T, Karafit M, Burget L, et al.Recurrent neural network based language model. Proceedings of the Conference of International Speech Communication Association. Chiba, Japan, pp.1045-1048, 2010.

[22] Sepp Hochreiter, Jrgen Schmidhuber.Long Short-Term Memory. Neural Computation, vol.9, no.8, pp.1735-1780, 1997.

[23] Richard S. Sutton and Andrew G. Barto. Reinforcement Learning:An Introduction,The MIT Press.

[24] Szepesvari C. Algorithms for Reinforcement Learning.vol.4, no.1, pp.632-636, 2010.

[25] R. S. Sutton and A.G. Barto, Reinforcement learning: An Introduction. Cambridge, MA, USA: MIT Press, 1998.

[26] R. S. Sutton and A.G. Barto, Introduction to Reinforcement Learning. Cambridge, MA, USA: MIT Press, 1998.

[27] G. Tesauro. TD-Gammon, a self-teaching backgammon program, achieves master-level play, Neural Comput., vol. 6, no.2, pp.215-219, 1994.

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

[29] http://tushare.org/index.html.

[30] Kingma D P, Ba J. Adam: A Method for Stochastic Optimization.Computer Science, 2014.