2018 11th International Symposium on Computational Intelligence and Design

Learning to Trade with Deep Actor Critic Methods

Jinke Li, Ruonan Rao
Shanghai Jiao Tong University
Shanghai, China
Email: {ljkxyoc, rnrao}@sjtu.edu.cn

Jun Shi
Shanghai Rongshi Investment Management Co. Ltd.
Shanghai, China
Email: shijun@rsfortune.com

***Abstract*—In this paper, we propose a new trading framework to apply deep actor critic methods to financial trading problems. Different from traditional actor critic methods, our model use not only actor but also critic to make the final decision. And for generalization purpose, a siamese structure in which the actor and the critic share the same LSTM features extraction part is adopted. The extracted features are then passed to different dense connected networks to compute q values and policy logits. The experiment results on different periods of CSI 300 prove that DACT has significantly better performances than B&H, DQT and DDRT. Furthermore, the idea of exploring based on the ensemble of actor and critic is valuable for other reinforcement learning problems.**

***Keywords*-Actor Critic; Deep Reinforcement Learning; Financial Trading;**

I. Introduction

For a long time, people pursue the valuable information hidden in the financial data. As a result, many mathematical models are proposed to describe the market environment and explain various market phenomenas. With these models, traders manage to develop strategies to earn excess profit based on their sophisticated financial knowledge. However, these strategies always do their jobs in short time, and rapidly lose effectiveness as time goes by.

To solve this problem, machine learning methods start to come up on the stage. The patterns digged by machine learning models depend on the data fed, and get updated when new data are provided. Therefore trading strategies based on machine learning methods show a far more promising future. Although trading with traditional machine learning models have made a great progress, it requires elaborative and well designed features. And as we all know, the deep learning (DL) approach automaticall extracts useful features through the hierarchical and deep networks, which makes end to end learning possible and sets human free from the harsh and tedious features design work [1]. Thus, many researchers adopt deep neural networks to improve the trading strategies and get higher profits.

However, as far, most of the trading strategies are based on the equity price prediction ([2]; [3]), which means the model predicts whether price will go up or down in the next period, and then the long, neural or short decisions are made. Prediction based method does not take the transaction cost and the previous actions into account, which are critical for online trading problems. Different from prediction based methods, the reinforcement learning (RL) methods use experiences gained through interacting with the environment and evaluative feedbacks to improve the ability to make decisions [4], which is naturally suitable for trading problems. [5] proposed a recurrent reinforcement learning (RRL) algorithm to do the online trading. And to have better function approximations and extract more effective features, deep reinforcement learning (DRL) which combines deep learning and reinforcement learning is put in the spotlight. [6] explain the traditional recurrent reinforcement learning process as a one-layer recurrent neural network and enhance it with a deep network. In fact, such methods can be categorized as a simplified type of deterministic policy methods without value functions. But the main advantage of deterministic policy gradient methods is solving problems in which the action space is infinite, and as for the trading problem, only long, neutral and short decisions need to be selected, so the more stable and well founded policy gradient methods with value functions, which are called actor-critic methods, should be chosen.

In our paper, we propose a new trading framework called deep actor critic trading (DACT). To take the order of financial data into account, we adopt recurrent neural networks (RNN), more precisely long short-term memory (LSTM) networks to extract features and do the function approximations; to generalize better, the LSTM network is shared by the value network and the policy network; to stablize the decision, the outputs of the two networks are taken the simple internal bagging on. The remaining parts of this paper are organized as follows. Section 2 generally reviews some related works about the DL, RL and DRL. Section 3 introduces the detailed implementation of the DACT model. Section 4 for DACT learning. Section 5 is the experimental part where we will verify the performance of DACT compared with other trading systems. Section 6 concludes this paper and indicates some future directions.

II. Related Works

Reinforcement learning is learning how to map situations to actions so as to maximize a numerical reward signal. The learner is not told which action to take, but instead must discover which action yields the most total return by inter-

2473-3547/18/$31.00 ©2018 IEEE
DOI 10.1109/ISCID.2018.10116

acting with the environment. According to use of different functions, reinforcement learning methods are divided into value-based methods and policy-based methods.

Value-based methods come up very early, such methods find functions mapping states or state action pairs to values which calculated with rewards based on the bellman equation. After learning, every time the agent can choose the action which maximizes the state action value. Dynamic programming methods are very useful when the full information of the environment can be got. But in most cases, we lack some information, so the monte carlo methods which are model free are put forward. And to accelerate learning, temporal-difference learning such as Q-learning and Sarsa which bootstrap are widely used. [7] used sharpe ratio as the objective to train the trading system based on Q-learning. [8] proposed an approach to perform pricing and selection for stocks with multiple Q-learning agents.

Different from value-based methods, policy-based methods learn a parameterized policy that can select actions without consulting a value function. A value function may still be used to learn the policy parameter, but is not required for action selection. [5] proposed the recurrent reinforcement learning model designed to learn risk controllable and cost sensitive trading strategies which use the sharpe ratio as the objective function. RRL uses deferred returns and recurrent trade position signals as input to the trading system. Since then, many researchers do their studies based on RRL trading framework. [9] compare RRL with the genetic programming method using daily, weekly and monthly opening prices of the S&P 500 stock index and report that RRL works better on higher frequency data. [10] merge threshold-based regime-switching models with RRL, and use the GARCH-based volatility measure as the transition variable.

While the RRL defines a good trading model, it is merely a shallow learning method, which is not capable for high dimension problems. As an emerging technique, deep learning takes over the complex and high dimension machine learning problems with distributed and hierarchical data representations. It makes a breakthrough in many fields such like computer vision [11], speech recognition [12] and natural language processing [13]. The combination of deep learning and reinforcement learning has become a fast-growing trend. [14] propose deep Q-learning networks (DQN) to make the performance in Atari2600 game increase more than 100%. [15] create AlphaGo which defeated the Go world champion. DRL has also achieved remarkable success in the computer vision [16] and natural language processing [17]. Therefore, researchers try to apply deep reinforcement learning to trading problems. [6] improve RRL with a deep network to automatically extract useful financial features. However, these methods do not use both value function and policy function. In our work, we use both value function and policy function to apply the deep actor critic methods to trading problems.

III. Deep Actor Critic Trading

A. Critic and Q-learning

In a reinforcement learning problem, an agent in a state s_t selects an action a_t at every time step t according to a policy $\pi(a_t|s_t)$, which maps states to actions, receives a reward r_{t+1} and gets transited to a new state s_{t+1} according to the environment dynamics. Our goal is to maximize the accumulated rewards, which is called the return $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$, where γ is the discount rate, $0 \leq \gamma \leq 1$. It's easy to know that the agent is more farsighted with larger γ, and vice versa.

Q-learning as an off-policy temporal difference method [18], updates the state action values, i.e., the expected returns according to Bellman optimality equation [19]

$$q_*(s,a) = \sum_{s',r} p(s',r|s,a)[r + \gamma max'_a q_*(s',a')] \quad (1)$$

where $p(s',r|s,a)$ represents the environment dynamics. The real environment usually can not be modeled, so based on Eq.1, at every time step t, the temporal difference is used to update the state action values

$$\begin{aligned} q(s_t,a_t) \leftarrow{} & q(s_t,a_t) + \alpha \\ & * [r_{t+1} + \gamma max_a q(s_{t+1},a) - q(s_t,a_t)] \quad (2) \end{aligned}$$

where α is the learning rate.

Here, we define $p_1, p_2, ..., p_t...$ as the price sequences of the trading equity. At every time step t, an action $\delta_t \in \{short, neutral, long\} = \{-1, 0, 1\}$ is taken, and the agent gets the profit $p_{t+1} - p_t$, so take the transaction cost c into account, the reward received is

$$r_{t+1} = \delta_t(p_{t+1} - p_t) - c|\delta_{t+1} - \delta_t| \quad (3)$$

Combine Eq.2 and Eq.3, we can do iterative updates to approximate the state action value $q_*(s,a)$ of the optimal policy.

B. Actor and Policy Gradient

Policy gradient methods learn a parameterized policy that can select actions without consulting a value function [20]. Assume θ_p is the policy's parameter, $J(\theta_p)$ is the performance measure, we need to maximize returns using gradient ascent: $\theta_{p,t+1} = \theta_{p,t} + \alpha \nabla J(\theta_{p,t})$. With some math, we can derive the policy gradient theorem

$$\nabla J(\theta_p) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\theta_p) \quad (4)$$

where μ is the on-policy state distribution under the policy π. And for the continuing problem like trading, the proportion is actually 1. Take A_t and S_t as expections under π, a step forward with Eq.4, the policy can be updated as:

$$\theta_p \leftarrow \theta_p + \alpha \mathbb{E}_\pi \left[q(S_t,A_t) \nabla_{\theta_p} ln \pi_{\theta_p}(A_t|S_t) \right] \quad (5)$$

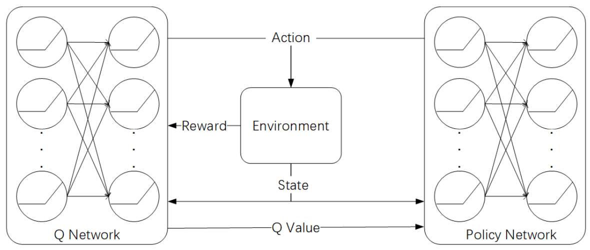

Figure 1. The Deep Actor Critic Trading model

C. Actor Critic with Deep Networks

Combine Q-learning and policy gradient, we can derive the actor critic method [21], in which q value as a critic is used to update the policy function, i.e., the actor.

Since the trading state space is infinite, we need to update the q value with function approximations as well, and this is when the deep networks take over. As revealed by Fig. 1, two deep networks are adopted to approximate the policy and value functions. It's crucial to notice that not only the policy network makes the decision but also the q network participates the decision-making process. For any state s_t, say that the decision of the policy network is A_t^p, and the decision of the q network is A_t^q, then the final decision is made by the following internal bagging function Eq. 6

$$A_t = \begin{cases} -1 & A_t^p + A_t^q < 0 \\ 0 & A_t^p + A_t^q = 0 \\ 1 & A_t^p + A_t^q > 0 \end{cases} \tag{6}$$

IV. DACT Learning

A. Siamese LSTM

Consider that the financial data are most time-series data, so RNNs are adopted to extract intrinsic features of the trading environment. However, simple RNNs are difficult to learn long-term dependencies, thus we turn to LSTMs, which keep useful cell states with input, forget and output gates in the long-term [22], to solve the challenge.

Although deep networks greatly empower the ability of function approximations, how to generalize well to new data is always a tricky problem. Think deeper of our model, the actor and the critic actually can share the same feature extraction part, and then make decisions through their own networks. As depicted in Fig. 2, the LSTM network which is responsible for capturing useful information of the trading environment is shared by the Q network and policy network. And then the extraced features are fed into two different dense connected networks to produce q values and policy logits respectively, which are further utilized to make the actor and critic decisions. The intuition behind is the Occam's razor principle, simpler is better, which means the less variables, the more generalization in our case, and such structure is called siamese network.

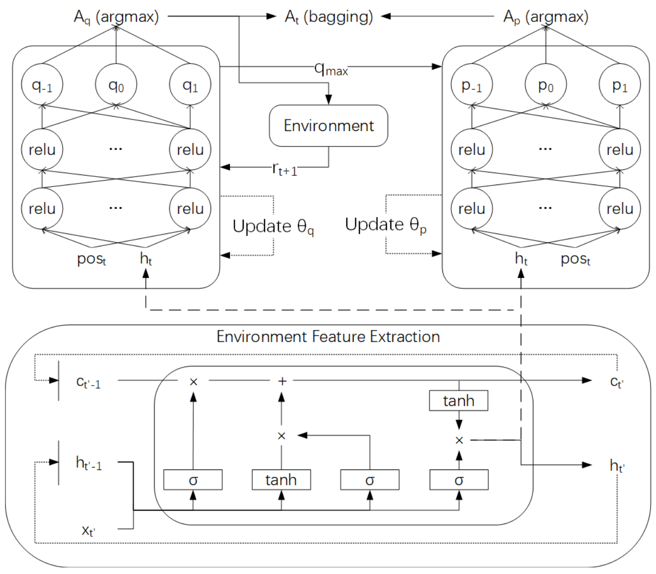

Figure 2. DACT learning detail. $X_t = \{x_1 ... x_{t'} ... x_t\}$ as the input environment state, $h_{t'}$ as the hidden state, $c_{t'}$ as the cell state, they recurrently produce the final state h_t, which is fed into decision networks together with pos_t, the position state. And then q_{pos} as the q value when the position state is $pos \in \{-1, -1, 0\}$, and p_{pos} as the policy logit are derived.

B. Backpropagation Through Time

To train the agent, we first need to figure out the way to update θ_p and θ_q. Eq. 5 illustrate how to update θ_p for the policy function. As for the q function, similarly, we can apply the semi-gradient update

$$\theta_q \leftarrow \theta_q + \alpha \mathbb{E}_\pi[(R_{t+1} + \gamma max_a q(S_{t+1}, A) \\ - q(S_t, A_t))\nabla_{\theta_q} q(S, A)] \tag{7}$$

Let's put aside other parts and focus on the way to update θ_l for the LSTM network now, $i_{t'}$ as the input gate, $f_{t'}$ as the forget gate, $o_{t'}$ as the output gate, $\tilde{c}_{t'+1}$ as the candidate cell state, the derivatives respect to $h_{t'}$ and $c_{t'}$ can be calculated as following

$$\frac{\partial c_{t'}}{\partial i_{t'}} = \tilde{c}_{t'+1} \quad \frac{\partial c_{t'}}{\partial f_{t'}} = c_{t'-1} \quad \frac{\partial c_{t'}}{\partial \tilde{c}_{t'+1}} = i_{t'} \quad \frac{\partial c_{t'}}{\partial c_{t'-1}} = f_{t'} \\ \frac{\partial h_{t'}}{\partial o_{t'}} = tanh(c_{t'}) \quad \frac{\partial h_{t'}}{\partial c_{t'}} = o_{t'}(1 - tanh^2(c_{t'})) \tag{8}$$

Next, assume L is the loss passed to LSTM, since we only have h_t, i.e., the final state, to backpropgate errors, so gradients at time t are $\frac{\partial L}{\partial h_t}$ and $\frac{\partial L}{\partial c_t} = \frac{\partial L}{\partial h_t}\frac{\partial h_t}{\partial c_t}$, and following equations for $t' < t$

$$\frac{\partial L}{\partial c_{t'}} = \frac{\partial L}{\partial h_{t'}}\frac{\partial h_{t'}}{\partial c_{t'}} + \frac{\partial L}{\partial c_{t'+1}}\frac{\partial c_{t'+1}}{\partial c_{t'}}$$

$$\frac{\partial L}{\partial h_{t'}} = \frac{\partial L}{\partial o_{t'+1}}\frac{\partial o_{t'+1}}{\partial h_{t'}} + \frac{\partial L}{\partial i_{t'+1}}\frac{\partial i_{t'+1}}{\partial h_{t'}} + \frac{\partial L}{\partial f_{t'+1}}\frac{\partial f_{t'+1}}{\partial h_{t'}} + \frac{\partial L}{\partial \tilde{c}_{t'+1}}\frac{\partial \tilde{c}_{t'+1}}{\partial h_{t'}} \quad (9)$$

With Eq. 5, Eq. 7, Eq. 8 and Eq. 9, we can update θ_p, θ_q and θ_l now.

C. Experience Replay without Sampling

In traditional actor critic method, the agent is offen following $\epsilon - greedy$ policy to explore the optimal policy, but there are only three actions (short, neutral, long) in our trading problem and we actually know the reward and the deterministic state after the action is taken during training, so sampling is useless in this particular phenomena. Specifically, every time in a state s_t, all three actions are taken in the environment to achieve three different state s_{t+1}^{-1}, s_{t+1}^{0}, s_{t+1}^{1}, which are used as experiences to replay later to update parameters.

V. Experimental Verifications

A. Experimental Setup

We test DACT model on the real-world financial data. One of the most important stock index of China market, CSI 300, is selected to evaluate the effectiveness. CSI 300 complied by the China Securities Index Company and calculated since 08/04/2005, is a weighted index representing the performance of the top 300 stocks of Shanghai and Shenzhen stock exchanges. Daily data till 08/04/2018 are collected to experiment on.

To demonstrate the robustness of our model, we distribute the data into three periods, 08/04/2005-08/04/2010, which goes quick upward and downward movements and then picks up, 08/04/2009-08/04/2014, which shows a typical downward trend struggling with resistances, and 08/04/2013-08/04/2018, which is similar to the first one, but more abruptly in the beginning and more moderate in the following. These periods are representatives of different market patterns.

Daily open, high, low and close subtracting previous close point are prepared as the features $x_{t'}$, and the sequence comprising of the last 60 day (about 1 trading season) features is fed into the trading system as X_t. The transaction fee c is set to 0.1 point which is much larger than the corresponding stock index future transaction fee. The discount rate γ is set to 0.9. The current 20 days (about 1 trading month) are used as test set, previous 60 days are used as validation set, 250 days (about 1 trading year) before the previous 60 days are used as train set, and every time the best one among 100 epochs is selected to apply to the test set, then the window slides forward 20 days.

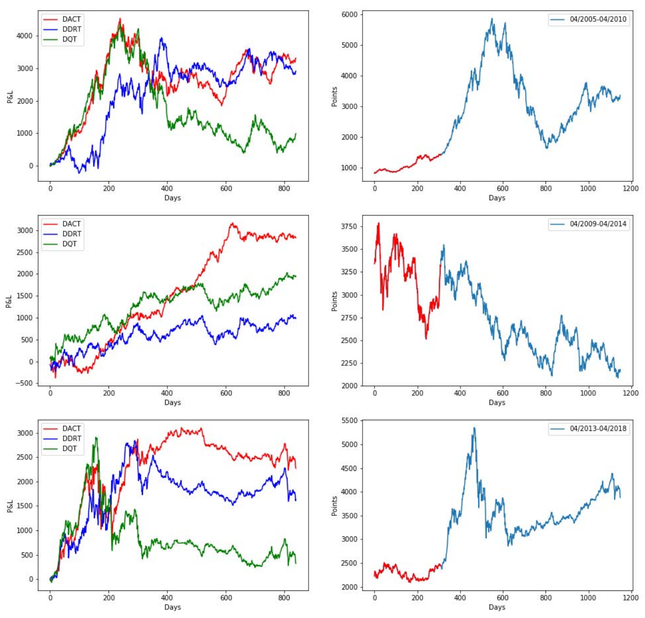

Figure 3. Points(right) and the P&L curves (left) of different trading models. Red parts are the initial training and validation data.

B. General Evaluations

To evaluate the effectiveness of our DACT model, we compare it with deep direct reinforcement trading (DDRT) system [6], which is a well improved version of RRL with the help of deep dense neural networks, deep Q trading (DQT) system as well as the simple buy and hold (B&H) strategy.

The results are shown in the Fig. 3. First it can be observed that our DACT model outperforms other competitors in all tests. DACT generally makes profits whether in upwards or downwards markets. Moreover, it adapts to new trends very quickly, which means it significantly reduces loss when the trend suddenly goes to the opposite and starts to make profits earlier in the new situation. The next observation is that DDRT performs nearly well as DACT in the first and third periods. However, it is inferior to both DACT and DQT in the downward movement. And it boots up slower than others in a upward movement. It is also too conservative in a fast switch market compared with DACT. And the third observation is that DQT only performs well in monotonic movements. It changes slowly when the trend changes.

Further more, Table. I gives the total profits (TP), sharpe ratios (SR) and position changes (PC) of different trading systems on CSI 300. Both DDRL and DACT beats B&H in all tests, and DQT only has advantage in the downward movement. DACT outperforms DDRT 17% in the first test period, 187% in the second test period and 45% in the third test period on total profits. DDRT only slightly exceed DACT in the first test period on sharpe ratios. And take

	04/2005-04/2010			04/2009-04/2014			04/2013-04/2018		
	TP	SR(%)	PC	TP	SR(%)	PC	TP	SR(%)	PC
B&H	1905	-	-	-1190	-	-	1459	-	-
DQT	955	5.3	98	1870	34.4	327	308	2.3	31
DDRT	2790	19.4	340	965	12.8	67	1531	14.3	252
DACT	3266	18.1	258	2771	40.1	289	2216	21.2	322

Table I
PERFORMANCE OF DIFFERENT TRADING SYSTEMS ON CSI 300

	keep_prob=0.1			keep_prob=0.5			keep_prob=0.9		
	TP	SR(%)	PC	TP	SR(%)	PC	TP	SR(%)	PC
num_units=64	149	1.2	138	1946	16.1	311	1896	19.1	312
num_units=128	895	6.5	97	2273	17.1	318	1719	13.6	337
num_units=256	825	6.4	324	3154	24.6	349	804	6.3	368

Table II
PERFORMANCE OF DIFFERENT HYPERPARAMETER SETTINGS

account of position changes in the table, DACT behaves more positively than DDRT when the trend changes.

Fig. 4 shows sample trading days of DACT. It can be observed that DACT takes actions nearly at bottom or peak, it indeed learned patterns and makes valuable decisions. DACT performs well whether in the downward movement (the left), or a consolidation following a upward movement (the right).

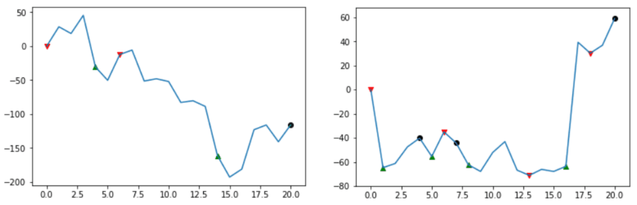

Figure 4. Sample trading days of DACT. Green triangle means long, red triangle means short and black circle means neutral.

C. *Hyperparameter Tuning*

Optimization and regularization are the central problems in machine learning. And good optimizations make the training error decrease quickly to the approximate minimum. Good regularizations make the test error close to the training error while still keep them as small as possible. Among many optimization aspects, we mainly focus on optimizers in this section. Adam optimizer not only dynamically adjust gradients for every parameter, but also integrates momentum to fast traverse narrow valleys [23]. Eq. 10 describes how it works, g_t as the gradient, $\eta = 0.001$ as the learning rate, $\beta_1 = 0.9, \beta_2 = 0.999$ as the exponential decay rates and $\epsilon = 10^{-8}$ for numerical stability.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \qquad v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$
$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \qquad \hat{v}_t = \frac{v_t}{1-\beta_2^t}$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \tag{10}$$

As for regularization, we use the simple but powerful technique, dropout, which can be regarded as a practical bagging method for deep neural networks [24]. Tab. II lists the results with different settings of number of units and keep probability tested on the third period data. We can see that all settings are underfitting when keep_prob=0.1, there is no wonder since only 10% of units take effect. And our model overfits when num_units=256 and keep_prob=0.9. Take the computation cost into consideration, we choose to set num_units=128 and keep_prob=0.5.

VI. Conclusions and Future Work

This paper applies deep actor critic method to financial trading problems. DACT updates parameters using experience replay without sampling during training, and uses not only actor but also critic to make the final decision during testing for stabilization. And to extract intrinsic features, LSTM networks are utilized to handle with financial sequential data. Moreover, the LSTM network is actually shared by the actor and critic networks in order to generalize better, and then following two different dense connected networks to compute q values and policy logits. Finally we demonstrate the effectiveness of our model by comparing it with B&H, DQT and DDRT experimented on different period of CSI 300, and DACT wins all tests.

Though DACT does really good compared with other trading systems, there are many aspects to explore and improve. First, we only use quotation data to trade, and there are many other data such as fundamental data (e.g. price/earnings ratio) and public opinion data (e.g. heat index) valuable for trading. We need to properly integrate them together to achieve better performance. Second, we simply combine the decisions of the q network and policy network for trading, but in other reinforcement learning problem where sampling is necessary, we actually can transform the q values and policy logits to some quantities under the same scale, then combine them to do the exploration and exploitation. Third, the market environment changes really fast and patterns learned from past is probably not applicable to current situation, and manually setting the period of validation data for online trading is not a carefully thought choice, the method to adapt quickly to new trends is still need to do further research.

References

[1] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[2] S. T. A. Niaki and S. Hoseinzade, "Forecasting s&p 500 index using artificial neural networks and design of experiments," *Journal of Industrial Engineering International*, vol. 9, no. 1, p. 1, 2013.

[3] J. Heaton, N. Polson, and J. H. Witte, “Deep learning for finance: deep portfolios,” *Applied Stochastic Models in Business and Industry*, vol. 33, no. 1, pp. 3–12, 2017.

[4] R. S. Sutton and A. G. Barto, “Reinforcement learning: An introduction (bibinfoedition2 ed.),” 2018.

[5] J. Moody and M. Saffell, “Learning to trade via direct reinforcement,” *IEEE transactions on neural Networks*, vol. 12, no. 4, pp. 875–889, 2001.

[6] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, “Deep direct reinforcement learning for financial signal representation and trading,” *IEEE transactions on neural networks and learning systems*, vol. 28, no. 3, pp. 653–664, 2017.

[7] X. Gao and L. Chan, “An algorithm for trading and portfolio management using q-learning and sharpe ratio maximization,” in *Proceedings of the international conference on neural information processing*, 2000, pp. 832–837.

[8] J. W. Lee, J. Park, O. Jangmin, J. Lee, and E. Hong, “A multiagent 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.

[9] D. Gorse, “Application of stochastic recurrent reinforcement learning to index trading,” in *ESANN*, 2011.

[10] D. Maringer and T. Ramtohul, “Threshold recurrent reinforcement learning model for automated trading,” in *European Conference on the Applications of Evolutionary Computation*. Springer, 2010, pp. 212–221.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[12] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,” *IEEE Signal processing magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[13] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” *arXiv preprint arXiv:1406.1078*, 2014.

[14] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing atari with deep reinforcement learning,” *arXiv preprint arXiv:1312.5602*, 2013.

[15] D. Silver and D. Hassabis, “Alphago: Mastering the ancient game of go with machine learning,” *Research Blog*, 2016.

[16] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, “Target-driven visual navigation in indoor scenes using deep reinforcement learning,” in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 3357–3364.

[17] J. Li, W. Monroe, A. Ritter, M. Galley, J. Gao, and D. Jurafsky, “Deep reinforcement learning for dialogue generation,” *arXiv preprint arXiv:1606.01541*, 2016.

[18] C. J. Watkins and P. Dayan, “Q-learning,” *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[19] R. Bellman, *Dynamic programming*. Courier Corporation, 2013.

[20] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” in *Advances in neural information processing systems*, 2000, pp. 1057–1063.

[21] A. G. Barto, R. S. Sutton, and C. W. Anderson, “Neuronlike adaptive elements that can solve difficult learning control problems,” *IEEE transactions on systems, man, and cybernetics*, no. 5, pp. 834–846, 1983.

[22] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[23] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[24] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.