2017 10th International Symposium on Computational Intelligence and Design

A Multi-Objective Deep Reinforcement Learning Approach for Stock Index Futures's Intraday Trading

Weiyu Si, Jinke Li, Ruonan Rao
School of Electronic Information and Electrical Engineering
Shanghai Jiao Tong University
Shanghai, China
{*swy015, ljkxyoc, rnrao*}*@sjtu.edu.cn*

Peng Ding
China Quantitative Investment Association
Shanghai, China
dingpeng@chinaqi.org

***Abstract*—Modern artificial intelligence has been widely discussed to practice in automated financial asserts trading. Automated intraday trading means that the agent can react to the market conditions automatically, while simultaneously make the right decisions. Besides, the profits will be made within a day considering transaction cost charged by the brokerage company. In this paper, we introduce a multi-objective deep reinforcement learning approach for intraday financial signal representation and trading. We design the deep neural networks to automatically discover the dynamic market features, then a reinforcement learning method implemented by a special kind of recurrent neural network (LSTM) is applied to make continuous trading decisions. In terms of balancing the profit and risk, we implement a multi-objective structure which includes two objectives with different weights. We conduct experiments on stock index futures data, and our analysis and experiments not only offer insights into financial market features mining, but also provide a straightforward and reliable method to make profits, which sheds light on its wide application on automated financial trading.**

***Keywords*-Deep learning, Reinforcement learning, Intraday trading, Financial signal processing.**

I. Introduction

The rapid development of artificial intelligence makes it possible to automatically find trading opportunities in the financial assert trading [1]. The process of trading can be described as a decision-making process, aiming to maximize the return while restraining the risk. Many researches have been done on machine learning approaches to financial market trading, but many of them try to predict price movement or trends [2], [3]. However, the prediction-based method is difficult to consider transaction cost, and the previous decisions.

Instead of price prediction, the reinforcement learning (RL) [4] uses experience gained through interacting with the environment and evaluative feedback to improve the ability to make decisions. These RL algorithms [5], [6], [7] directly output the discrete trading signals on financial market trading. However, the good performance of RL is based on the effective features from the environment. To reduce the noise and make stable decisions, deep reinforcement learning (DRL) is drawing much attention due to its remarkable achievements in AlphaGo [8] and playing atari games [9]. The concept of DRL is to use the deep learning to extract hidden features and the reinforcement learning to make decisions.

Using DRL can update the trading strategy according to the market environment and adjust the action based on the previous actions and market conditions to reduce the loss. Whereas at the same time, a lot of difficulties still exist in training DRL trading systems. One is that when we explore the unknown market environment with high noise and non-stable financial data, there is no supervised information for the system to learn. In addition, the system makes decisions when it explores the environment at the same time. Therefore, it is important to improve the continuity of the decision-making process while learning the strategy, and change the action according to the environment conditions, thus ensuring the robustness of decision-making process.

To address the two problems, in this paper, we implement a multi-objective deep reinforcement learning (MODRL) approach for simultaneously exploring the environment and recurrently making decisions for financial intraday trading. The MODRL includes two parts of neural networks. The first part is composed of four layers of fully-connected networks for learning features from the financial data and reducing the uncertainty of the input data. The second part is a layer of Long Short Term Memory (LSTM) and a layer of fully-connected network to implement self-taught reinforcement trading for decision making. In order to balance the profit and risk, we propose a multi-objective structure. It has two objectives to measure the profit and risk separately with different weights. The two objectives can be changed for different goals and better performance.

The rest parts of the paper are organized as follows. Section II reviews some related work about the RL, DL and DRL. Section III introduces the detailed implementations of the MODRL system. Section IV for MODRL learning. Section V is the experimental part where we will verify the performances with other trading systems. Section VI concludes this paper.

2473-3547/17 $31.00 © 2017 IEEE
DOI 10.1109/ISCID.2017.210

II. Related Work

Reinforcement Learning (RL) [4] is a widely studied mechanism of self-learning and is used to solve the Markov decision process [10]. According to the goal of different learning problems, typical reinforcement learning can be roughly divided into two categories, actor-based and critic-based models.

For actor-based reinforcement learning, it directly learns policy from the data and make continuous decisions directly from the policy. Moody and Saffell [5] proposed a recurrent reinforcement learning (RRL) model designed to implement risk control, cost-sensitive trading strategies that use the Sharp ratio as an objective function. RRL used a set of lagged returns and recurrent trade position signals as input to the trading system. Since then, various studies have been based on the RRL trading system [6], [11], [12]. The critic-based algorithm evaluates the value function. These value function methods, such as TD-learning [4], Q-learning [13] and Sarsa [14], are often used to solve discrete space optimization problems. In recent years, many researchers try to use Q-learning for financial trading [15], [16], [17].

While RL provides a good trading model, it is well known that robust features are important for decision-making. Poor feature extraction will directly affect the trading system. For extracting good features, the deep learning [18] behaves well in the image classification [19], speech recognition [20], and natural language processing [21]. Deep learning is a representation-learning framework, and its infrastructure is described as a multi-layers neural networks. Deep learning can learn robust features from the big data. The combination of deep learning and reinforcement learning has become a fast-growing trend in the robot control [22], computer vision [23], natural language processing [24] Some researchers try to use DRL to solve the financial trading [25]. Deng et al. [25] used the deep neural networks to extract abstract features from data and identify non-linear hidden relationships without relying on human experience or econometric assumptions.

III. Multi-Objective Deep Reinforcement Learning-MODRL

A. Recurrent Reinforcement Learning and Multi-Objective Structure

The basic recurrent reinforcement learning (RRL) is introduced by Moody et al. [26] and it is essentially a single-layer recurrent neural network. It can be described as a nonlinear function to approximate the trading action at each time step:

$$A_t = tanh(\langle w, f_t + b\rangle + uA_{t-1}) \tag{1}$$

As defined in Eq.1, $\langle\cdot,\cdot\rangle$ is the inner product, A_t represents the trading decision $\{long, neutral, short\}$ token at time t, and f_t represents the feature vector of the current market condition at time t, and (w, b) are the coefficients for the feature regression. $\Theta = \{w, b, u\}$ is a family of parameter set that is trained to maximum the trading objective. In RRL, the action at time t not only depends on the current market condition at time t, but also relies on the previous action at time $t-1$, because the frequently changing trading positions will bring large transaction costs.

In basic RRL, the price sequence is described as $p_1, p_2, ..., p_t, ...$, and the return at time t is defined by $r_t = p_t - p_{t-1}$. The feature vector at time point t uses lagged returns $f_t = [r_{t-m+1}, ..., r_t] \in R^k$, which means the recent m return values are directly composed of the feature vector at time t. $A_t \in \{1, 0, -1\}$ represents the trading decision $\{long, neutral, short\}$. The profit R_t made by the trading model is defined by

$$R_t = A_{t-1}r_t - c|A_t - A_{t-1}| \tag{2}$$

The first term is the profit or the loss obtained based on the decision A_{t-1} at time $t-1$ and the second term is the transaction cost when changing the trading positions at time t. The transaction cost is needed when two consecutive decisions are different.

The most straight RRL objective is to maximize the accumulated rewards in the period $1, 2, ..., T$. The accumulated rewards throughout the whole T training period can be defined as

$$\max_{\Theta} U_T\{\sum_{t=1}^{T} R_t|\Theta\} \tag{3}$$

Another RRL objective is Sharp Ratio (SR), which is the risk adjusted returns. The Sharp ratio is defined as the ratio of average return to standard deviation of the returns obtained in period $1, 2, ..., T$, i.e.

$$SR = \frac{mean\{R_1...R_T\}}{std\{R_1...R_T\}} \tag{4}$$

With the objective (Eq. 3 or Eq. 4) and the trading policy (Eq. 1), the optimization of RRL is to learn the parameter set Θ that maximize the objectives in Eq. 3.

In this paper, for intraday trading, our objective is to maximum the average daily profits and reduce the trading risk. Therefore, we introduce a multi-objective structure can be described as

$$U = \alpha \cdot mean\{DR_1...DR_{TT}\} - \beta \cdot std\{DR_1...DR_{TT}\},$$

$$DR = \sum_{t=1}^{k} R_t \tag{5}$$

Where k represents the period of a day, and the first term is to measure the average daily profits, and the second term is to describe the volatility, which means the smaller it is, the smaller risk is taken during the trading. The parameter α, β is adjustable to give the system direction what kind of strategy the system will learn. In this paper, $\alpha = 1, \beta = 0.01$.

B. Deep Reinforcement Neural Networks

Deep learning is a powerful feature learning framework that has been demonstrated in many fields [27]. In our implementation, we use multiple fully-connected hidden layers (FC) to extract hidden features. We define the deep transformation as $F_t = g(f_t)$, which extracts abstract features from the input vector f_t through a nonlinear mapping $g(\cdot)$. In each FC layer, each node on the $(l+1)$ layers is connected to all the nodes in the lth layer. If in_i^l represents the input of the ith node on the lth layer then out_i^l is its corresponding output.

$$\begin{aligned} in_i^l &= w_i^l \cdot out^{l-1} + b_i^l \\ out_i^l &= \begin{cases} in_i^l, & in_i^l > 0 \\ 0, & else \end{cases} \end{aligned} \tag{6}$$

where the parameters $\{w_i^l, b_i^l\}, \forall i$ are the adjustable parameters to be learned.

After learning the hidden features, we implement a special kind of recurrent neural network to make decisions. In this paper, we use the long short-term memory (LSTM) networks [28], which is designed to solve the long term dependencies. LSTM keeps the trading actions in the memory to generate the successive actions and reduce the transaction cost. And, the LSTM can remember more features over long time steps to make good decisions. Then, the trading action is subject to the following equation:

$$\begin{aligned} i_t &= \sigma(w_{xi} \cdot x_t + w_{hi} \cdot h_{t-1} + b_i) \\ f_t &= \sigma(w_{xf} \cdot x_t + w_{hf} \cdot h_{t-1} + b_f) \\ o_t &= \sigma(w_{xo} \cdot x_t + w_{h0} \cdot h_{t-1} + b_0) \\ g_t &= tanh(w_{xg} \cdot x_t + w_{hg} \cdot h_{t-1} + b_g) \\ c_t &= f_t \otimes c_{t-1} + i_t \otimes g_t \\ h_t &= o_t \otimes tanh(c_t) \end{aligned} \tag{7}$$

The features and the previous outputs are saved in the memory cell c_t. At each time step, forget gatesf_t determines how much the memory should be let through, and the input gatesi_t generates the new information to be added to the memory cell. Lastly, the output gate o_t and new memory cell state c_t generate the result.

The last layer is a fully-connected hidden layer, with the $tanh$ activation function. The range of the output of the last layer is -1 to 1, which represents trading decisions. h_t is the output of the LSTM layer. The parameters $\{w, b\}$ are the parameters to be learned.

$$A_t = tanh(w \cdot h_t + b) \tag{8}$$

In conclusion, the overview of MODRL for feature learning and decision making is shown in Figure 1. The raw data is the input of the feature learning part, then abstract features from the feature learning part are used to make decisions in the decision making part. Lastly, the decisions are directly the outputs of the trading system.

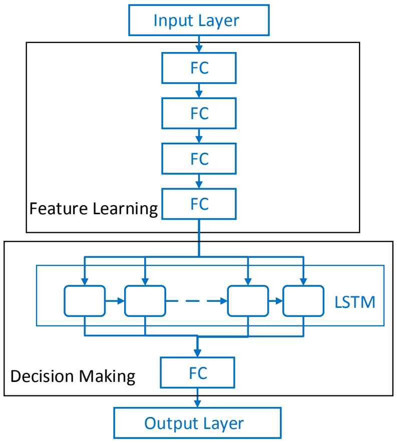


Figure 1. Overview of MODRL for feature learning and decision making

IV. MODRL Learning

As we all know, deep neural networks involve thousands of latent parameters to be learned. Vanishing gradients and overfitting are the most common problems when training the deep neural networks. In this section, we introduce practical strategies to train MODRL.

A. Dropout for Feature Learning

Dropout is the most popular regularization technique for training deep neural networks. It is designed to avoid overfitting in the training process. At each training step, every neuron, excluding the output neurons, has a probability p of being ignored during this training step, but it may be active during the next training step. The hyperparameter p is called $dropout\ rate$. For ease of explanation, one hidden layer is specified here. We define x as the input of the ith node on the lth layer, $\tilde{x}$ as the input of the $i$the node on the lth layer after dropout, and y_i^l is its corresponding output. With the dropout architecture, the network can be describe as:

$$\begin{aligned} D_i^l &\sim Ber(p) \\ \tilde{x}_i^l &= D^l \star x^l \\ y_i^l &= f(z_i^l) \\ z_i^l &= w_i^l \tilde{x}^l + b_i^l \end{aligned} \tag{9}$$

Where $\star$ represents the element-wise product and D is a matrix of independent Bernoulli Ber (p) distributed random variables.

The dropout strategy is only used during training process. After training, neurons don't get dropped anymore. In this paper, the dropout rate is set to 50%. and the dropout is used on the input layer and four FC layers to avoid over fitting features. The LSTM layer and the last output layer will not use dropout because it is to make decisions recurrently.

B. *MODRL for Trading*

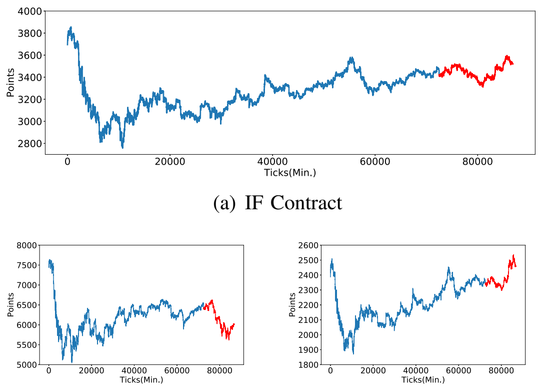
We use Long Short Term Memory (LSTM) to reduce the gradient vanishment and remember long term information. For intraday trading, the number of time steps unfolded in LSTM is the same as the minute-level trading time within a day. In China, the trading time is 4 hours within a day, so the time steps is 240. We force to close the position at the last minute, which means that the last action token within a day is 0.

The output of the MODRL neural networks gives the direction for trading signals, and the trading signal with high confidence was trusted. We define a parameter δ to distinguish between three actions, as shown in Eq. 10. In this paper, $\delta = 0.5$.

$$y = \begin{cases} 1, & p > \delta \\ 0, & else \\ -1, & p < -\delta \end{cases} \tag{10}$$

V. Experimental Results

A. *Experimental Setup*



(a) IF Contract

(b) IC Contract

(c) IH contract

Figure 2. Prices of the three tested future contracts. Blue parts: train data. Red parts: test data.

We test the MODRL trading system on the real-world financial data of stock index futures. We select the stock-IF contract, stock-IH contract and stock-IC contract, which are three index-based future contracts traded in China. All these contracts exhibit very high liquidity, and allow both short and long operations. When the subsequent market price goes highter (respectively, lower), the long (respectively, short) position makes profits.

We use the minute-level close prices, which means that the interval between p_t and p_{t+1} is 1 min. The historic data of the contracts in the minute solutions are shown in Figure 2. We use eighteen months data and the data period ranges from 2016-01 to 2017-06. It can be found that these three contracts include many different market patterns, especially in test data (red parts). IF data gets very large upward and downward movements in the tested period. The IC contract, generally, shows a downward trend and the IH contract has no obvious direction at first and shows a upward trend in the testing period.

In practical, the transaction cost is an important factor of intraday trading. The inherent value of these contracts is evaluated by China Yuan (CNY) per point (CNY/pnt). As for IF, the increase (decrease) in one point lead to a reward of 300 CNY for a long (short) position. The transaction cost (TC) charged by the brokerage company needs to be considered. In the experiment, we set TC$= 1$.

The lagged return with 200 return values is directly used as the input of the trading system. The four FC layers has 100, 100, 50, 50 hidden nodes per layer. Then the feature representations of the final FC layer are connected with the LSTM layer with 20 hidden nodes. Lastly, the output of the LSTM layer is the input of the last FC layer with one hidden node for trading decision-making.

B. *General Evaluations*

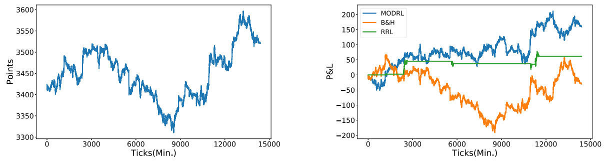
In this section, we evaluate the MODRL system on practical data. The system is compared with the Recurrent Reinforcement Learning (RRL) [26] system and Buying and Holding (B&H) Strategy.

The training set is the data from 2016.01˜2017.03, and the test data is the data from 2017.04˜2017.06. The maximum training epoch of MODRL is 5000 epochs, and the adaptive moment estimation (Adam) [29] optimization is adopted to automatically adjust the network parameters and learning rate. As for the RRL system, the code is provided by Yuou Song and Kartik Shetty [1]. We modify the code to adapt to intraday trading. In the RRL system, the Sharp Ratio is the objective and the limitation is to hold a position at least five time steps. The input features are the lagged return with 120 return values, and the maximum training epoch is 100.

The details of the profit and loss (P&L) curves and price curves of test data are shown in Figure 3, Figure 4 and Figure 5. The quantitative evaluations are summarized in Table I where the performance is reported in the forms of AP, SR and TT. As for intraday trading, we use average profits (AP) gained each day to analyze, and TT is the total trading number.

In Figure 3, MODRL outperforms RRL and B&H on total profits. MODRL make profits both on upward trend and downward trend. In most cases, RRL choose to do nothing, which may due to the volatility within a day when considering the transaction cost. B&H follows the trend of

[1] https://github.com/MagicEthan/CS534 AI Proj

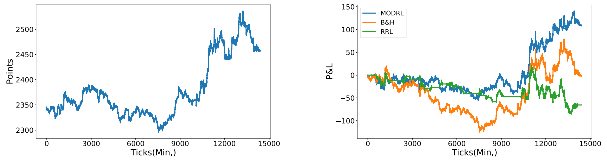


(a) IF Contract

(b) Performances

Figure 3. IF testing data and the P&L curves of different trading systems

close price each day. B&H make profits when the open price is lower than close price within a day.

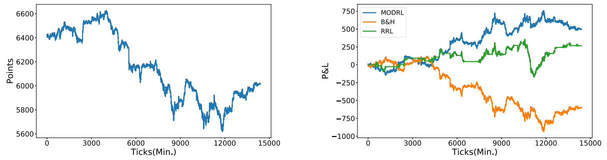


(a) IH Contract

(b) Performances

Figure 4. IH testing data and the P&L curves of different trading systems

In Figure 4, it is obvious that when the trend is not obvious, RRL, MODRL and B&H do not make profits, but in the latter, MODRL gains better performance than B&H and RRL. The RRL performs worse than B&H, which may be because RRL pay more transaction cost for changing positions.



(a) IC Contract

(b) Performances

Figure 5. IC testing data and the P&L curves of different trading systems

From Figure 5, MODRL and RRL achieve much more profits during the trading period. MODRL are more stable to gain profits, and the RRL fluctuates during some periods. B&H has the same trend as the prices, and makes loss when the price is downward trend.

It can be observed from Table I that, for the intraday trading, MODRL can gain more average points each day than RRL, which means MODRL is possible to make profits for intraday trading. As for Sharp Ratio, MODRL obtains the higher SR than RRL and B&H, except for IF contracts, because the total trading number of RRL is only 6. MODRL has the highest number of transactions and makes the highest profits in these three trading systems.

From the experimental results, it can be found that feature learning can contribute to better performances, and the LSTM network makes the continuous decisions and change positions at the right time, which reduces the transaction cost. Another observation we can found is that the multi-objective structure makes good profits within acceptable risk.

VI. Conclusion

This paper introduces the deep reinforcement learning into the typical RRL framework for financial signal processing and trading. The MODRL uses the deep neural networks to find the effective features from a large amount of financial data. The reinforcement learning is achieved by a recurrent network, which uses the previous action as the feedback in order to make continuous actions. In order to balance the profits and risks, the two objectives with different weights are used, and the two objectives can be changed for solving different problems or better performance. The results on stock-index future contracts demonstrate that the effectiveness of MODRL in finding features of market condition and action learning which exhibits both statistical and economical significance.

References

[1] E. W. Saad, D. V. Prokhorov, and D. C. Wunsch, “Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks,” *IEEE Transactions on neural networks*, vol. 9, no. 6, pp. 1456–1470, 1998.

[2] 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.

[3] 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.

[4] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

	IF			IH			IC		
	SR	AP	TT	SR	AP	TT	SR	AP	TT
B&H	-0.023	-0.51	-	-0.0028	-0.43	-	-0.16	-10.03	-
RRL	0.73	1.02	6	-0.0097	-1.09	46	0.083	4.44	64
MODRL	0.12	2.66	62	0.11	1.8	80	0.11	8.30	91

Table I

PERFORMANCES OF DIFFERENT TRADING SYSTEMS ON DIFFERENT CONTRACTS

[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] M. A. Dempster and V. Leemans, “An automated fx trading system using adaptive reinforcement learning,” *Expert Systems with Applications*, vol. 30, no. 3, pp. 543–552, 2006.

[7] J. Cumming, D. Alrajeh, and L. Dickens, “An investigation into the use of reinforcement learning techniques within the algorithmic trading domain,” 2015.

[8] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, “Mastering the game of go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[10] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

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

[12] D. Maringer and T. Ramtohul, “Threshold recurrent reinforcement learning model for automated trading,” *Applications of Evolutionary Computation*, pp. 212–221, 2010.

[13] G. Tesauro, “Td-gammon: A self-teaching backgammon program,” in *Applications of Neural Networks*. Springer, 1995, pp. 267–285.

[14] S. Singh, T. Jaakkola, M. L. Littman, and C. Szepesvári, “Convergence results for single-step on-policy reinforcement-learning algorithms,” *Machine learning*, vol. 38, no. 3, pp. 287–308, 2000.

[15] 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.

[16] 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.

[17] X. Du, J. Zhai, and K. Lv, “Algorithm trading using q-learning and recurrent reinforcement learning,” *positions*, vol. 1, p. 1, 2016.

[18] Y. Bengio *et al.*, “Learning deep architectures for ai,” *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[19] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, “Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations,” in *Proceedings of the 26th annual international conference on machine learning*. ACM, 2009, pp. 609–616.

[20] A. Graves, A.-r. Mohamed, and G. Hinton, “Speech recognition with deep recurrent neural networks,” in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 2013, pp. 6645–6649.

[21] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *arXiv preprint arXiv:1409.0473*, 2014.

[22] S. Levine, N. Wagener, and P. Abbeel, “Learning contact-rich manipulation skills with guided policy search,” in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 156–163.

[23] 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.

[24] 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.

[25] 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.

[26] J. Moody, L. Wu, Y. Liao, and M. Saffell, “Performance functions and reinforcement learning for trading systems and portfolios,” *Journal of Forecasting*, vol. 17, no. 56, pp. 441–470, 1998.

[27] T. Ohyama, W. L. Nores, J. F. Medina, F. A. Riusech, and M. D. Mauk, “Learning-induced plasticity in deep cerebellar nucleus,” *Journal of Neuroscience*, vol. 26, no. 49, pp. 12 656–12 663, 2006.

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

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