Reinforcement Learning Strategies in Algorithm Stock Trading

Tao Jin
jin.hybr@gmail.com

***Abstract*—Algorithm stock trading is the core part of financial technologies and how to develop efficient and profitable trading strategies is vital to all traders and financial institutions. This paper investigates the different Deep Reinforcement Learning strategies and proposes a reinforcement learning model to solve the algorithm trading problem.**

***Index Terms*—Deep Reinforcement Learning, Markov Decision Process, Algorithm Trading, Portfolio Management**

I. Introduction

The financial industry has evolved dramatically over the last decades and continues to face more and more challenges amid rapidly increased competitions, technologies developments and complex political and economic environments. To face these challenges, how to apply Machine Learning methods to develop the profitable winning strategies in the algorithm stock trading and portfolio management, framed in terms of risk-factor exposure, dynamic environment, as opposed to asset classes, is becoming a more important problem. The success of Deep Learning has achieved a lot of state of the art performances in many industries, from computer vision, natural language processing to autonomous driving and robotics application. At the same time, reinforcement learning also has gained a lot of attention from academic communities to various industries, combined with the techniques from deep learning, Deep Reinforcement Learning have created an intelligent agent to interact with an unknown dynamic environment and aimed to maximize its cumulative rewards. By using the methods from deep learning, Deep Reinforcement Learning is learning through trial and error from simultaneously sequential and evaluative feedback and sampled by nonlinear function approximation. Based on these characteristics, Deep Reinforcement Learning is a promising approach in algorithm trading and portfolio management.

II. Problem Statement and Task Definition

Algorithm stock trading is the core part of financial technologies and how to develop efficient and profitable trading strategies is vital to all traders and financial institutions. The target for stock trading is to maximize the returns based on the estimated potential return and risks. Currently, the major problems of algorithm trading is:

A. Highly transaction cost based on all dynamic factors in the markets

Based on Markowitz's mean-variance optimization, Given the expected return, variance and co-variance matrix, the optimization process revolves around finding the asset weights that will generate the portfolio with highest Sharpe ratio, so the traders or portfolio managers should make decisions every time based on the information changed, the process is complex and costly [1].

B. State Spaces Problem

The other approach is to use dynamic programming and Markov Decision Process (MDP) to design the optimal strategies [2]. However, in the stock market this model is limited when the number of state spaces is growing larger due the market is dynamically changing. In [3], the author applied the different Reinforcement Learning configurations into the trading system, but limitation is the management of learning rate based on stationary systems. For the dynamic real time financial data, the system can't handle the increasingly state spaces. In contrast to supervised learning in solving algorithm trading and portfolio management, Reinforcement Learning does not require creating the rules under which a certain action must be considered true with a certain weight and allows using the metrics calculated for each strategy for long time intervals, for instance, the Sharpe ratio. [4] To tackle these problems, this project will investigate how to apply deep reinforcement learning methods and design effective strategies (trading policies - determining the optimal trading positions:buy, hold or sell) to solve these challenges.

III. Literature Review

Traditional research on algorithm trading is majorly focused in are the trend following and mean reversion strategies, which are covered in detail in [9] and [10]. With the success of Machine Learning (ML) applications in the various industries, in the algorithm trading field, the key ML models are Financial Time Series(FTS) data forecasting. From all the ML models and algorithms, the recurrent machine learning algorithms tend to perform better at the task of financial market prediction than simple feed forward models, presumably due to their ability to take into account FTS' temporal dynamics. The hypothesis for ML model forecasting is that if the price evolution is predictable in advance with a certain level of confidence, the trading policy can be easily calculated. In [11], The authors used Convolution Neural Network(CNN) price direction predictor to design high frequency trading strategies. [12] used wavelet transforms, stacked auto-encoders and long short-term memory (LSTM) to compute the trading policy.

The key differences between FTS forecasting and Algorithm Trading is the performance metrics. The forecasting models prefer to use the goodness of fit measures such as Root-Mean-Squared-Error(RMSE), but in algorithm trading, the models always use either Sharpe Ratio or the rate of return. Reinforcement Learning(RL) is a natural solution to some finance and economics problems [5], like option pricing, and multi-period portfolio optimization, where value function based RL methods were used. RL was first introduced and implemented in the financial market in 1997 [13]. Since then there are a lot of accomplishments, for example: In [6], authors proposed to utilize policy search to learn to trade; [7] extended policy search with deep neural networks. Deep (reinforcement) learning would provide better solutions in some issues in risk management. The market efficiency hypothesis is fundamental in finance. To handle the behavioral biases in human decision-making under uncertainty, in [7] the authors proposed Cumulative Prospect Theory which is applied to RL in financial data prediction.

There are several main RL methodologies in algorithm trading field:

A. *Base Reinforcement learning*

Using the base RL Model method, the key is to optimise a financial portfolio in order to maximise the return over a long period. The algorithm was model free but the parameters were learned and iteratively improved using Policy Gradient and Actor-Critic Methods on real past stock market data [14]. Whenever the algorithm took an action, the net change in the portfolio was received as feedback in order to either reinforce or discourage the previous trading decision made. The disadvantages are the result of Sharpe Ratio is only 0.486.

B. *On and Off Policy Reinforcement Learning*

RL can be divided into two separate categories: On policy learning and Off policy learning with algorithms trained using these categories in [15], [16] They were both online policies in that they were learning new information to make decisions when they were running at the end of each step, as opposed to offline policies that only learned once the algorithm finished running at the end of the final step such as Support Vector Machine . On-Policy learning included State Action Reward State Action(SARSA) that learned policies and evaluated consequences of the action currently taken. Off-Policy learning included Q-learning policies and would evaluate the rewards independent of the current action as they were always evaluating all the possible current actions to see which action could maximise the reward gained at the next step.The hypothesis of both Q-Learning and SARSA is based on the market not able to adopt the new information as quickly as it arrived. The limitation for Off-Policy [15] is not considered the transaction cost which is the key constraint in algorithm trading.

C. *Combining Recurrent Neural Network and Q-Learning*

In [15], the authors had proved the benefits of introducing the non-linear relationships into complex neural networks to represent the financial market trading events. In [17], authors combined the Q-learning algorithm with the deep neural network, between the FTS data input and output action.

D. *Combining Deep Neural Network and Recurrent RL*

The main work of this method is described in [18]. The authors proposed a Markov Decision Process(MDP) model that was solved using deep recurrent Q-network and sampled a longer sequence for Recurrent Neural Network(RNN) learning. It included a few training steps and extra feedback to the agent to eliminate the need for random exploration during RL. However, this technique was only applicable to stock trading under a few market assumptions including the cost of any transaction was a fix percentage of the value of foreign exchange currency traded. It also required increases in sequence sampling for RNN training which reduced the time interval required for training the algorithm. The log of daily returns was defined as the reward function. The advantage was a larger transaction spread generated better returns suggesting this was likely due to the constraint imposed by the system that forced the algorithm to look for more creative ways to profitably trade on the market. A major disadvantage was the short time period for all currency pairs over the same period of time between 2012 and 2017 which diluted the advantages of using multiple currencies. Based on the literature reviews, this project will focus more on the MDP model and combine with Deep Q-learning Network and RNN methods.

IV. Related Work

The key RL approaches for discrete or continuous financial markets are Actor-Only, Critic-Only and Actor-Critic approach.

A. *Critic-Only Approach*

The Actor-Only learning is that the agent directly learns the optimal policy itself. The Neural Network only learns the policy. To solve the state space problem and improve the trading efficiency, this approach introduces Recurrent RL [6]. The Actor-Only learning can only handle the continuous action space environment [20].

B. *Actor-Only Approach*

The Critic-Only learning is to solve the discrete action space problem by using Deep Q-learning Network(DQN). Q-Value function is to learn the optimal policy that maximizes the expected future rewards given the current states. DQN uses a Neural Network to perform the function approximation about Q-Value. The limitation for this approach is stock trading is a continuous state space and Critic-only method is only can train a single stock or asset [20].

C. Actor-Critic Approach

The Actor-Critic learning [23] has applied in financial markets recently. The advantages for this method is that Actor network and Critic network can be simultaneously updated. The Actor network updates the trading policy's probability distribution guided by the critics policy gradients while the Critic network updates the value function. So the Actor learns to take better actions and Critic learns better policy [21] [22].

V. Algorithm Trading Problem Formalization

The key target of algorithm trading is to maximize the portfolio values. The algorithm should be based on the states(fundamental data, alternative data and portfolio values) to decide what, when, how and at which price to make decision of trading actions - **Buy, Sell or Hold**. So the whole algorithm trading can be considered a complex sequential decision making process. We can define the algorithm trading problem as Markov Decision Process(MDP).

The elements of this process are at the time stamp t, the portfolio value v_t and the trading state is S_t of the time t. The trading action a_t stands for - **Buy, Sell or Hold**. So the final target is to seek the best trading action a_t from the policy $\pi(a_t, s_t)$.

A. Hypotheses

- Time Horizon Discretization
  Stock Market Trading can be treated a continuous process during the trading time. The simplest way to solve a continuous-state MDP problem is to discretize the state space, then we can use the reinforcement learning algorithm to solve the problem. we can discretize the the continuous of trading time horizon into the high number of trading time stamps t, the trading action a_t is based on the policy $\pi(a_t, s_t)$
- Trading Pipeline
  The dynamic trading activities can be viewed a stochastic model. The pipeline for choosing the right action at the time stamp t based on the following steps:
  1. Update the trading State S_t based on market information and portfolio values;
  2. Calculate the Policy $\pi(a_t \mid s_t)$ function to decide the best action a_t;
  3. Execute the action a_t;
  4. Shift the time stamp from t to $t+1$
- States vs. Information
  In the Policy function π will maximize an optimal policy which is decided by immediate and future rewards over the a certain time horizon. In the Reinforcement Learning definition, the optimal policy π^* is defined as the following:

$$\pi^*_\theta = \underset{\theta}{\operatorname{argmax}} E(R) \mid \pi_\theta) \tag{1}$$

$$R = \sum_{t=0}^{\infty} \gamma^t r_t \tag{2}$$

  γ is the discount factor which is decide the future reward weights in the whole reward function.
  The states is based on the market information and value changes due to the action **Buy, Sell or Hold**. The limitations for algorithm trading model is
  1. The market information is very limited to model
  2. The form of information is various (tableau data, text etc.)
  So in the model, need add more constrains to reduce the bias which is bought by limited information.

B. Data

We will use the Dow Jones 30 constituent stocks at(01-01-2016) as base trading pool. The training data set will be from 01-01-2009 to 01-01-2015. The validating data set will be from 01-02-2015 to 12-31-2015. Finally the model will be evaluated by the data from 01-01-2016 to 06-30-2020.

- Data Prepossessing and Augmentation
  The purpose of Data Prepossessing is to remove the noise. The normalization and transformation will be used in processing the raw price data. Data augmentation is the key part for generating more synthetic data for training. The model will use signal shifting, signal altering and noise addition.

C. Metrics

The target of algorithm trading is to maximize the profit. The metrics for this model will be use the Sharpe Ratio, a performance indicator widely used in the portfolio management. Mathematically, the Sharpe ratio definition S_r:

$$S_r = \frac{E[R_p - R_f]}{\sigma_r} = \frac{E[R_p - R_f]}{\sqrt{var[R_p - R_f]}} \tag{3}$$

where:
R_p is the expected return
R_f is the risk free return
σ_r is the standard deviation of the portfolio

Also the model will calculate the annualized Sharpe ratio as a reference

$$S_annual = \frac{S_r}{\sqrt{252}} \tag{4}$$

where:
252 is the number of trading days in a year.

D. Model

The Algorithm Trading model will be defined by a MDP model, which includes States, Policy, Action, Reward function and Value Function.This Decision process can be defined as MDP = $[S, A, P, \gamma, R]$, where

- States: $S = [P_t,\ N_t, C_t,\ I_t]$, $S \in \mathbf{R^m}$ is the space of Observed States;
  where:
  For each time stamp t, the observed states is $s_t \in S$. Inside the vector s_t, P_t is the prices vector at current time stamp t in the portfolio. N_t is the vector of shares numbers of each stock at time t. C_t is the cash value

of the current portfolio. I_t is the information indicator vector. These indicators provide the more insights of market trends, such as Moving Average Convergence Divergence.(MACD), Relative Strength Index(RSI), Average Direction Index(ADI) and Average Directional Movement Index(ADX).

- Action A = the spaces of actions Sell, Hold or Buyfor the stocks. At the each time stamp t, the action $a_t \in A$ is decided from the policy $\pi(a_t \mid s_t)$ function, that is the probability to choose a_t from s_t.
- Policy $P = (s_{t+1} \mid s_t, a_t)$, which is represented the probability distribution of action a_t at the state s_t
- $R(s, a)$ is the reward function, for time stamp t, $r_t = R(s_t, a_t)$
- $\gamma \in [0, 1]$, the total reward for action a_t,

$$R_t = \sum_{i=0}^{T} \gamma^i r_{t+i} \tag{5}$$

The objective of the algorithm is to find the strategy $\pi : S \to A$ that maximizes the $E[R \mid \pi(\theta)]$.

E. General Reasoning and Algorithm

Based on the MDP model, Actor-Critic Algorithm [23] will be applied in this problem. The benefits of Actor-Critic algorithm are both Actor network and Critic network can be simultaneously updated. The Actor network updates the trading policy's probability distribution guided by the critics policy gradients while the Critic network updates the value function. So the Actor learns to take better actions and Critic learns better policy [21] [22].

The pipeline for **Actor-Critic algorithm** includes three key parts:

- Run the Policy and generate the samples: sample the (s_i, a_i) from $\pi_\theta(\mathbf{a} \mid \mathbf{s})$
- Fit a Deep Neural Network model to estimate return, to fit $\widehat{V}^\pi$ to sample reward sums
- Evaluate

$$\widehat{A}_\pi(s_i, a_i) = r(s_i, a_i) + \widehat{V}^\pi_\phi(s_i^{'}) - \widehat{V}^\pi_\phi(s_i) \tag{6}$$

- The Objective Function

$$\bigtriangledown_\theta J(\theta) \approx \sum_i \bigtriangledown_\theta \log \pi_\theta(a_i \mid s_i) \widehat{A}_\pi(s_i, a_i) \tag{7}$$

- Update the policy

$$\theta \leftarrow \theta + \alpha \bigtriangledown_\theta J(\theta) \tag{8}$$

To avoid the high variances in estimated advantage values, this algorithm use the a critic network to estimate the use of rewards to go. So we can rewrite the $\widehat{A}_\pi(s_i, a_i)$ as follow:

$$\widehat{A}_\pi(s_i, a_i) \approx r(s_t, a_t) + \gamma \widehat{V}^\pi_\phi(s_{t+1}) - \widehat{V}^\pi_\phi(s_t) \tag{9}$$

For updating the value function network, define the following target values:

$$y_t = r(s_t, a_t) + \gamma \widehat{V}^\pi_\phi(s_{t+1}) \tag{10}$$

Then setting the regression objects to optimize the gradient descent:

$$\min_\phi \sum_{i,t} (\widehat{V}^\pi_\phi s_{i,t} - y_{i,t})^2 \tag{11}$$

VI. Algorithm Implementation

A. Stock Data Processing

The stock data set will use the Dow 30 historical data. The data elements include the daily Open(prcod), High(prchd), Low(prcld) and Close Prices(prccd); the daily volume for common share traded(cshtrd), the Cumulative Adjustment Factor Issue for the stock (AJEXDI). The Adjusted Closing Price will be used for the stock price which is compute by $\frac{PRCCD}{AJEXDI}$.

Based on these fundamental data information, the algorithm will build stock price indicator vector that includes the popular price trend indication information into the State Space. By using the *stockstats* python package to compute the MACD, RSI ,ADI and ADX indicator.

So for one single stock t will be include P_t (Adjusted Closing Price) and Information vector
$I_t = (MACD_t, RSI_t, ADI_t, ADX_t)$

B. Trading Environment Setting - States Space

As the model chooses Dow 30 as the trading pool, so $m = 30$. Based on Trading States Definition:

$S = [P_t,\ N_t, C_t,\ I_t]$, $S \in \mathbf{R^{30}}$. For any time t, the model has the following elements:

- $P_t \in \mathbf{R^{30}}$, the Dow 30 Stocks price vector at time t;
- N_t, for each Dow 30 stock, the holding share numbers at time t;
- C_t, at the time t, the Dow 30 portfolio balance (Cash Value);
- $I_t = (MACD_t, RSI_t, ADI_t, ADX_t)$, for each Dow 30 stock, its related trends indicator vector at time t;

So the total Dow 30 portfolio trading states spaces will be 30(P_t) + 30(N_t) + 1(C_t) + 4*30(I_t) = 181.

C. Trading Environment Setting - Action Space

For each stock in the Dow 30 portfolio, the action will be buy, sell or hold. The model will set the N_{max} as maximum share to trade. So at the time t, the Dow 30 action spaces will define as the following:

For each trading action, the number shares of buying will be positive integer number $k, k \leq N_{max} = 100$, the number of selling will be negative integer number $-k, k \leq N_{max}$. The action space $\mathbf{A} = \{-k, ..., -1, 0, 1, ...k\}$.

So the entire action space for Dow 30 portfolio will be $(2k{+}1)^{30}$. Based on the implementation of OpenAI [24], THE

Actor-Critic algorithm is based on a Gaussian distribution, the action space vector need to be normalized and symmetric from original vector.

D. Reward Function, Policy and Actor-Critic Model

The goal for trading algorithm is to maximize the Dow 30 portfolio's positive cumulative cash value. At the time t, the portfolio value is C_t, based on the policy $(s_{t+1} \mid s_t, a_t)$, at the the new state $s_t + 1$, the trading algorithm should maximize the portfolio value:

$$r(s_{t+1} \mid s_t, a_t)) = (C_{t+1} + P_{t+1}^T N_{t+1}) - (C_t + P_t^T N_t) - Cost_t \tag{12}$$

$Cost_t$ is the transaction cost. Normally, the cost rate will set to $r = 0.1\%$

$$Cost_t = P^T * r \tag{13}$$

For time $t + 1$, the current shares N_{t+1} should equal the current share numbers plus buying numbers then minus the selling numbers. The same as reward function value.

$$R(s_{t+1} \mid s_t, a_t)) = R_{hold} - R_{sell} + R_{buy} - Cost_t \tag{14}$$

More generally, we define the reward function based on the action $\bar{a}, \bar{a} \in \{hold, sell, buy\}$.

$$R_{\bar{a}} = (P_{t+1}^{\bar{a}} - P_t^{\bar{a}})^T N_t^{\bar{a}} \tag{15}$$

For maximizing portfolio value, the algorithm should maximize R_{hold} and R_{buy} and minimize R_{sell}. For $\gamma \in [0, 1]$, the objective of the Actor-Critic Model is to find the strategy $\pi : S \to A$ that maximizes the $E[R \mid \pi(\theta)]$. The Objective function $\bigtriangledown_\theta J(\theta)$ is as Equation 7.

E. Validation

After training, the algorithm will use the Sharpe Ratio as the metric to evaluate performance. To simplify the Sharpie Ratio equation 3. The algorithm will use the multiple month validation rolling window to compute the ratio. The Sharpe Ratio can be defined:

$$Sharpe\ Ratio = \frac{R_{expected} - R_{risk\_free}}{\sigma_{expected}} \tag{16}$$

F. Implementation

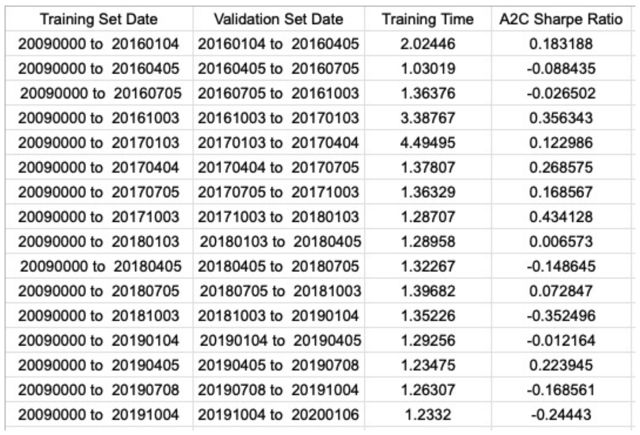
Based on OpenAI gym environment, the algorithm uses the A2C model(Actor-Critic Model) from openAI's stable baseline [24] to build the trading strategy and train the agent.

Algorithm 1: Reinforcement Trading Algorithm

```
Result: Best Trading Strategy to maximize the Dow
        30 Portfolio value
1. Processing Stock data;
2. Building the train/valid data set by adding
   price,trends indicators and etc. information ;
3. Environment Setting: Parameters setting,
   Train/Trade/Valid Environment(build states,action) ;
4. Initialization Environment;
5. Training $model = A2C('MlpPolicy', env\_train)$
while timestamps do
    Train the A2C model ;
    model.learn()
end
6. Validation Model ;
7. Compute the Sharpe Ratio ;
```

G. Result

The algorithm trains Dow 30 stock data. The Training data set starts from 2009.01.01, the rolling windows for validation is three-month. The basic results from the algorithm is:



Training Set Date	Validation Set Date	Training Time	A2C Sharpe Ratio
20090000 to 20160104	20160104 to 20160405	2.02446	0.183188
20090000 to 20160405	20160405 to 20160705	1.03019	-0.088435
20090000 to 20160705	20160705 to 20161003	1.36376	-0.026502
20090000 to 20161003	20161003 to 20170103	3.38767	0.356343
20090000 to 20170103	20170103 to 20170404	4.49495	0.122986
20090000 to 20170404	20170404 to 20170705	1.37807	0.268575
20090000 to 20170705	20170705 to 20171003	1.36329	0.168567
20090000 to 20171003	20171003 to 20180103	1.28707	0.434128
20090000 to 20180103	20180103 to 20180405	1.28958	0.006573
20090000 to 20180405	20180405 to 20180705	1.32267	-0.148645
20090000 to 20180705	20180705 to 20181003	1.39682	0.072847
20090000 to 20181003	20181003 to 20190104	1.35226	-0.352496
20090000 to 20190104	20190104 to 20190405	1.29256	-0.012164
20090000 to 20190405	20190405 to 20190708	1.23475	0.223945
20090000 to 20190708	20190708 to 20191004	1.26307	-0.168561
20090000 to 20191004	20191004 to 20200106	1.2332	-0.24443

VII. Conclusion

This algorithm has verified the A2C model is easy to build trading model, without other tools and metrics, this model can manage the 181 state spaces and $(2 \times 100 + 1)^{30}$ action space(100 is the max trading share number). The limitation is the model need more trading constrains to improve the Sharpe ratio. Also the trading algorithm need to integrate other models, such as Proximal Policy Optimization (PPO) or Deep Deterministic Policy Gradient (DDPG) to improve the results.

References

[1] Yong Zhang and Xingyu Yang, “Online portfolio selection strategy based on combining experts’ advice,” Computational Economics, vol. 50, 05 2016.

[2] Dimitri Bertsekas,Dynamic programming and optimal control,vol. 1, 01 1995.

[3] Francesco Bertoluzzo and Marco Corazza, “Testing different reinforcement learning configurations for financial trading: introduction and applications,” Procedia Economics and Finance, vol. 3, pp. 68–77, 12 2012.

[4] W. F. Sharpe, “The sharpe ratio,” J. Portfolio Management. 21, 49–58, 1994.

[5] Hull, J. C. Options, Futures and Other Derivatives (9th edition). Prentice Hall, 2014.

[6] Moody, J. and Saffell, M. Learning to trade via direct reinforcement. IEEE Transactions on Neural Networks, 12(4):875–889.2001.

[7] Deng, Y., Bao, F., Kong, Y., Ren, Z., and Dai, Q. Deep direct reinforcement learning for financial signal representation and trading. IEEE Transactions on Neural Networks and Learning Systems.2016.

[8] Prashanth, L., Jie, C., Fu, M., Marcus, S., and Szepesari, C. Cumulative prospect theory meets reinforcement learning: Prediction and control. In the International Conference on Machine Learning (ICML). 2016

[9] Chan, E. P. Algorithmic Trading: Winning Strategies and Their Rationale. Wiley. 2013.

[10] Narang, R. K. Inside the Black Box. Wiley. 2009.

[11] Arevalo, A., Ni no, J., Hernandez, G., and Sandoval, J. (2016). High-Frequency Trading Strategy Based on Deep Neural Networks, ICIC. 2016.

[12] Bao, W. N., Yue, J., and Rao, Y. A Deep Learning Frame work for Financial Time Series using Stacked Autoencoders and Long-Short Term Memory. PloS one, 12. 2017.

[13] Moody, J.; Wu, L.; Liao, Y.; Saffell, M. Performance functions and reinforcement learning for trading systems and portfolios. J. Forecast. , 441–470.1998, 17.

[14] Kanwar, N. Deep Reinforcement Learning-Based Portfolio Management; The University of Texas at Arlington, TX, USA, 2019.

[15] Pendharkar, P.C.; Cusatis, P. Trading financial indices with reinforcement learning agents. Expert Syst. Appl., 103, 1–13. 2018.

[16] Marco Corazza, A.S. Q-Learning and SARSA: A comparison between two intelligent stochastic control approaches for financial trading. Univ. Ca’ Foscari Venice Dept. Econ. Res. Pap. 15, 1–23.2015.

[17] Jeong, G.; Kim, H.Y. Improving financial trading decisions using deep Q-learning: Predicting the number of shares, action strategies, and transfer learning. Expert Syst. Appl. 117, 125–138.2018.

[18] Huang, C.-Y. Financial Trading as a Game: A Deep Reinforcement Learning Approach. Arxiv Quant. Finance.arXiv:1807.02787.

[19] Zhengyao Jiang and Jinjun Liang, “Cryptocurrency portfolio management with deep reinforcement learning,” in 2017 Intelligent Systems Conference, 09 2017.

[20] Lin Chen and Qiang Gao, “Application of deep reinforcement learning on automated stock trading,” in 2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS), pp. 29–33.2019.

[21] Zhengyao Jiang and Jinjun Liang, “Cryptocurrency portfolio management with deep reinforcement learning,” in 2017 Intelligent Systems Conference, 09 2017

[22] Yuxin Wu and Yuandong Tian, “Training agent for first-person shooter game with actor-critic curriculum learning,” in International Conference on Learning Representations (ICLR), 2017, 2017.

[23] Vijay Konda and John Tsitsiklis, “Actor-critic algorithms,” Society for Industrial and Applied Mathematics, vol. 42, 04 2001

[24] Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave,Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford,John Schulman, Szymon Sidor, and Yuhuai Wu, “OpenAI: Stable baselines,stable-baselines.readthedocs.io”