Reinforcement Learning to Trade the Stock Market

Zarin Subah Shamma, and Nicholas S. Flann

***Abstract*—Uncertainty associated with the potential return from stocks closely aligns with the change in the company's stock price. This non-linear, arbitrary, and uncontrollable fluctuations pose to be one of the main reasons behind low long-term returns from stock purchases. Investment companies often use stock trading strategies which are challenging and not optimal for a complex and dynamic stock market. In order to make the right choices when it comes to stock trading, Reinforcement Learning has the potential to overcome the shortcomings humans have in terms of making trading decisions and foreseeing long-term stable profit. In this paper, the potential of reinforcement learning has been explored to optimize stock trading strategy which maximizes investment return in the long term. Two agents are trained with historical stock data using two different learning algorithms; Deep Q-learning and Actor-Critic reinforcement learning. The agent with maximum investment return after trading for 5-months with unseen data identifies deep q-learning to be the best trading strategy among them with a minimal amount of training data.**

***Key Words*—Close price, reinforcement learning, stock, deep q-learning, actor-critic, profit, loss, reward.**

I. Introduction

One of the most common concerns of financial analysts and investors is making accurate predictions about stock prices [1], but it is also a difficult task for them [4]. This is because the majority of stock prices are highly volatile. Numerous different types of factors influence stock prices. Direct economic indicators such as fluctuations in consumer supply and demand and commodity price indexes are complex enough. Stock price changes have become increasingly elusive as a result of the global environment and investor's behavior [11]. Each factor generates forces in distinct directions, which eventually result in a change in the stock price. The correlation between various factors is frequently so obscure that it is difficult to identify the factors affecting stock prices, let alone build a model for stock price prediction on this basis.

For usual investors, stock analysis methods are usually hard to be taken into practice due to the information asymmetry of stock market, lack of trading experience and human weakness, etc [21]. Advances of reinforcement learning (Sutton and Barton 1998) can learn good policies for sequential decision problems and have outperformed human beings in many tasks such as playing Atari games [16]. So can we train an Reinforcement Learning (RL) agent to learn from trading experiences and make profitable trading policy automatically? Compared with common RL tasks, stock trading has a few difficulties to imply RL due to the following reasons:

Firstly, how to define the state of stock trading environment is crucial for the performance of RL trader. In tasks like playing Atari games, the state of the environment is defined as the last four continuous game screens and is a full representation of the environment. However, unlike the aforementioned deterministic and noise-free environment, the stock prices are known as full of noises and trading environment is changing over time [10]. As a consequence, we need to define a state that satisfies both of the following criteria:

- The state needs to contain as much information of the factors that affect stock prices as possible.
- The state needs to contain less noise so the agent can learn right experience and is more likely to choose right action at each time step so that a positive long-term accumulative reward can be acquired.

Secondly, since stock prices change continuously and there is no boundary of the prices, the state space can be extremely large [15], so it is hard to give an accurate estimation of the value of each action at each state with a relative limited dataset.

To address the aforementioned two problems, we have trained a deep reinforcement learning agent, namely Deep Q-Learning (DQN) Agent and an Actor-Critic Agent, to find profitable patterns in the complex and dynamic stock trading environment

and learn a better policy to achieve long-term accumulated profits.

The paper is organized as follows. First, the problem has been stated in terms of the stock trading in Section I. Next in Section III, the problem statement has been introduced. In section IV, the Deep Q-Learning Approach is described followed by the Actor-Critic Reinforcement Learning Approach in Section 1. After that, the experimental results are discussed and compared in Section VI. The paper concludes with a discussion of the potential impact of this work and the remaining challenges which need to be solved to maximize the long term profits even more in Section VII.

II. Related Works

In recent years, researchers have become increasingly interested in evolutionary algorithms like genetic algorithm [6], [8], [9], and artificial neural networks [7], to come up with stock trading strategy. Deep reinforcement learning has been applied to such areas as high frequency trading and investment portfolios in financial pair trading. To be more exactely, the Reinforcement Learning (RL) algorithm have been used in quantitative finance [3]. The superiority in applying RL concepts in finance is common, which includes the automated processing real-time data with high-frequency, and conduct transactions efficiently with the use of agent. For example, both Sarsa (On-Policy TD Control) and Q-learning (Off-Policy Temporal Difference Control Algorithm) are used by the optimization algorithm of optimized by JP Morgan Chase trading system [20]. League Champion Algorithm (LCA) [2] for the extraction stock trading rules, the process extracts and holds multiple stock trading rules for diversiform stock market environment.

Recent applications of deep reinforcement learning in financial markets consider discrete or continuous state and action spaces [12]. Learning models with continuous action space provide finer control capabilities than those with discrete action space. The actor-critic approach has been recently applied in finance [5], [13], [14], [24]. The idea is to simultaneously update the actor network that represents the policy, and the critic network that represents the value function. The critic estimates the value function, while the actor updates the policy probability distribution guided by the critic with policy gradients. Over time, the actor learns to take better actions and the critic gets better at evaluating those actions. The actor-critic approach has proven to be able to learn and adapt to large and complex environments, and has been used to play popular video games, such as Doom [23]. Thus, the actor-critic approach is promising in trading with a large stock portfolio.

III. Problem Statement and Approach

We model stock trading as a Markov Decision Process (MDP), and formulate our trading objective as a maximization of expected return.

- State s: The state is defined as difference in the close data in the given window. The difference has been taken so that we can determine the trend of the close prices.

$$s_t = [P_{t10} - P_{t9}, P_{t9} - P_{t8}, ..., P_{t2} - P_{t1}, P_{t1}] \quad (1)$$

- Action a: The action space only includes three available values [0, 1, 2], representing do nothing, buy and sell for one share of stock respectively. The action is chosen either randomly or using th greedy approach. We want to buy stocks in low price and sell stocks in higher price. To do so, whenever we the model wants to sell a stock, it compares the present close price with the buying price. It is a profit, it buys it, otherwise it holds it doing no change.
- Reward r: The reward is calculated as follows:

$$\text{Reward} = \frac{\text{Starting Money} - \text{Current Money}}{\text{Current Money}} \quad (2)$$

The starting money is the inventory that we want to invest in the trading. The current money will get changed whenever one stock will be bought or sold.

The state is used as the input of the networks with the size of the window and the output are the Q-values/probabilities for three different actions. In actor-critic network, the actor decided which action should be taken looking at the probabilities and critic inform the actor how good was the action and how it should adjust by adjusting and updating

the probabilities/q-values. The agent chooses the action with the highest value with a probability of $(1-\epsilon)$ and will get a reward. Then, the trade sample $\langle s_t a_t r_t s' \rangle$ will be stored in the buffer. Because the training dataset is rather small for a deep neural network and may cause overfitting. To enrich the data used for updating the network parameters and make the updating more stable, we actually store three samples with three different rewards into the buffer. After doing this, the accumulated reward is calculated and a number of batch-size samples will be taken randomly from the buffer to update the network parameters. The whole process will be carried forward till the end of the test data and an accumulated reward can be obtained.

IV. Deep Q-Learning Agent

Deep Q-learning is Q-Learning with the Q-table be replaced with a deep neural network. Mnih was the first to propose a deep Q-network (DQN) model by combining convolutional neural networks in deep learning and the Q learning algorithm in conventional reinforcement learning [17], [18]. This model incorporates a convolutional layer, which significantly improves the learning efficiency and performance [19].

This approach aims to deal with the problem that when the state space and action space are continuous or infinitely discrete, it becomes impossible to iterate the Q-value in Q-table with finite samples. Since it has been proved that a neural network with 3 layers is able to express any function like

$$Q'(s,a) = Q(s,a) + \alpha(Q^*(s,a) - Q(s,a)) \quad (3)$$

The main idea of deep Q-learning is to use a neural network to approximate the Q-function. The Q-network can be trained by minimizing the error between the improved Q-value given by formula

$$Q^*(s,a) = \mathrm{E}[r + \gamma \max_a Q^*(s^\vartheta,\ a^\alpha)] \quad (4)$$

and the original Q-value. The loss function can be minimized by the stochastic gradient descend (SGD) algorithm.

Although deep Q-learning performs well on many tasks, one well-known drawback of deep Q-learning is that it is quite unstable, especially in the noisy and time-varying stock trading environment. So one approach to make it stable is to use a separate target network $\tilde{Q}$ to compute the improved Q given by the Bellman equation:

$$Q(s,a) = r + \gamma \max_a Q(s',a') \quad (5)$$

DQN's training process makes use of the experience replay mechanism [22] (experience replay). The transferred samples $e_t = (s_t, a_t, r_t, s_{t+1})$. When an agent and environmental samples are transferred to a playback memory unit $D = (e_1, ..., e_t)$ at a given time t, they are stored there. Small batches of random transfer samples are selected from D each time, and the Stochastic Gradient Descent (the SGD) algorithm is used to update network parameter θ during training.

V. Actor-Critic Learning Agent

In the Actor-Critic approach, the actor's job is to select actions at each time step to form the policy, where the critic's role is to evaluate these actions taken by the actor. Figure **??** maintains an actor network and an critic network. The actor network maps states to actions and the critic network outputs the value of action under that state.

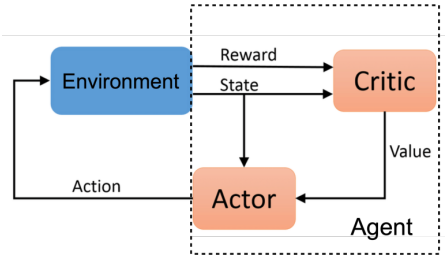

Fig. 1: Actor-Critic Reinforcement Learning Network. There will be two neural networks; one actor network and one critic network. The actor network selects actions at each time-step and the critic network evaluate these actions looking at the state and reward function and minimizing the loss.

So the approach is gradually adjusting the policy parameters θ of the actor to take actions that maximize the total reward predicted by the critic. The TD error (δ) calculated the critic to evaluate the action is as follows:

$$\delta = R_{t+1} + \gamma \hat{V}(s_{t+1}, \omega) - \hat{V}(s_t, \omega) \quad (6)$$

The value function estimation of the current state $\hat{V}(s_t, \omega)$ is added as a baseline to make the learning faster. The parameter θ of the actor is being adjusted in the way of maximizing the total future reward. The equation used by the Actor-Critic to update the gradient at each time step t as the following:

$$\theta_{t+1} = \theta_t + \alpha \Delta \ln \pi(a_t|st, \theta_t)(R_{t+1} + \gamma \hat{V}(s_{t+1}, \omega) - \hat{V}(s_t, \omega)) \tag{7}$$

Actor-critic network uses an experience replay buffer R to store transitions and update the model, and can effectively reduce the correlation between experience samples. Target actor-critic networks are created by copying the actor and critic networks respectively, so that they provide consistent temporal difference backups. Both networks are updated iteratively. At each time, the actor-critic agent takes an action a_t on s_t, and then receives a reward based on s_{t+1}. The transition (s_t, a_t, r_t, s_{t+1}) is then stored in replay buffer R. The critic network is then updated by minimizing the expected difference between outputs of the target critic network and the critic network. After the critic network and the actor network are updated by the transitions from the experience buffer, the target actor network and the target critic network are updated.

VI. Results

The data used in the experiments involves 2 years and 5 months ranging from 01/01/2016 through 05/31/2018. The data is divided into a training dataset (01/01/2016-12/31/2017) which is data of 2 years and a test dataset (01/01/2018-05/31/2018) which is data of 5 months. We firstly used the training data to initialize the networks by 2000 iterations. Then the performance of the models are tested on the test dataset.

A. Deep Q-Learning Agent

The deep Q-network contains 3 layers and the number of units for each layer is 5, 256 and 3 respectively. The Gradient Descent Optimizer has been used to minimize the average loss. For the Q-learning part, the discount factor is set to $\gamma = 0.95$, this value is quite small because the problem we defined is a short-term daily trading process so that we pay more attention to current reward. The max ϵ is set to 0.5 and minimum to 0.01 with a decay rate of 0.999. The batch size is set to 32 with a memory of 1000 length.

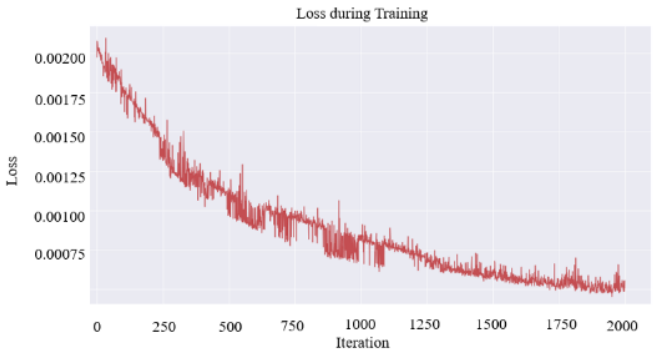

Fig. 2: Loss during Training the Deep Q-learning Agent. With the increasing number of iterations, it is clear that, the loss has decreased when the algorithm is getting trained with the the stock market data fluctuations.

We have trained the deep Q-learning agent with 2 years' Google stock market data. We have only considered the close prices (the last price at which a stock trades during a regular trading session) as described in Section III. In the agent, the average loss was minimized using the Gradient Descent optimizer. The loss curve shown in the figure 2 follows a decreasing trend which is highly expected. Before the 1750th iteration, the loss has decreased steadily with no massive increment. After the 1750th iteration, the loss becomes stabilized at around 0.00075 unit. This signifies that the model stopped learning after around 1750 iterations.

When the deep Q-learning agent was being trained, the profit gained from trading was partially stable and then there was a positive augmentation in the gained profit as in Figure 3. This indicates, up until around 1000 iterations, the agent was learning the trading policy at a slower rate which increased afterwards. It is noticeable that, at around 1750 iteration, the agent started to forget but it overcame this at around 2000 iterations.

For the testing, we have traded the data for 5 months and in the end, the profit is -65.75. After taking a look at the Figure 4, it can be determined what the agents is suggesting for selling or buying stocks. The usual practice is to buy the stock when

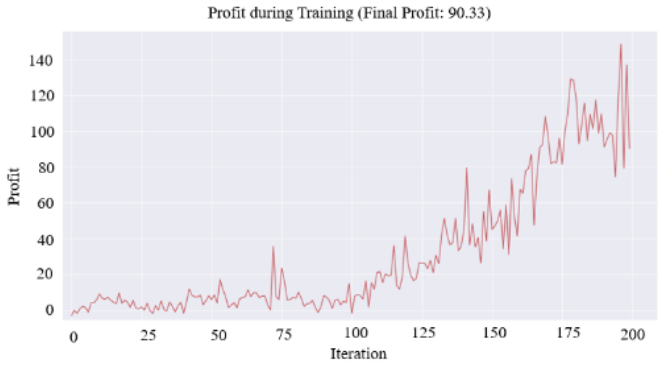

Fig. 3: Profit during Training the Deep Q-Learning Agent. We have recorded the data each 10 iterations. It is clearly seen that, up until 1000 iterations, the profit was stable and it increased drastically after that with a minor decline at around 1850th iteration.

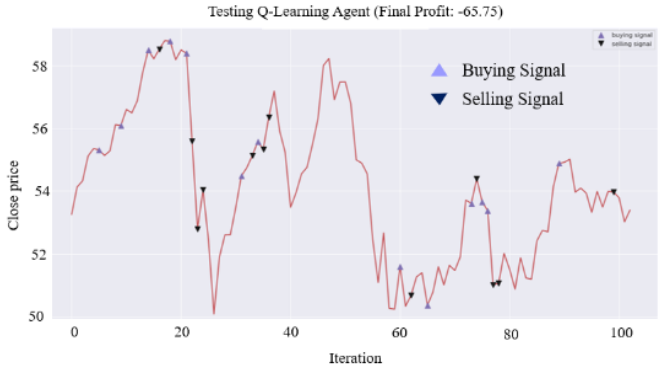

Fig. 4: Testing the Deep Q-Learning Agent with unseen data of 5-months. After trading in for 5 months, the final profit is -65.75.

the price is plummeting and sell it when it the price is higher. Having said that, the agent chose the right action by selling the stock at around 20. It made mistakes around 17 iteration as it was buying three stocks when the price was higher. Similarly, at around 60, the agent was right to buy the stocks but at the same time, it sold one which does not fall under the normal trend of buying and selling stock. The agent was at hold at around 50 whereas that would have been a potential time to sell the stock at a higher price as after that period, there is a drastic downfall in the stock price. But with this small amount of training data, the result is not bad to learn a good trading policy to maximize the return in the long run.

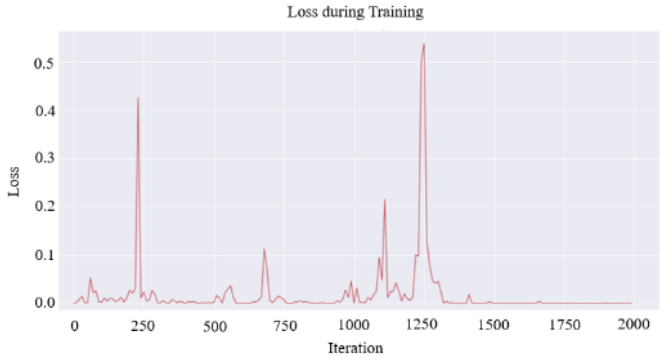

Fig. 5: Loss during Training the Actor-Critic Agent. The loss curve did not follow any trend. At some points, the loss is really high which defines the poor learning of the agent.

B. Actor-Critic Agent

The actor-network contains 3 layers and the number of units for each layer is 5, 256 and 3 respectively. The critic-network contains 6 layers and the number of units for each layer is 5, 256, 256, 128, 64 and 3 respectively. There is one LSTM-RNN cell in both of them. The Adam Optimizer has been used to minimize the average loss. The learning rate is set to be 0.001. The discount factor is set to $\gamma = 0.99$, this value is quite small because the problem we defined is a short-term daily trading process so that we pay more attention to current reward. The max ϵ is set to 0.5 and minimum to 0.1 with a decay of 0.005. The batch size is set to 32 with a memory of 1000 length.

There were two neural networks in the actor-critic agent. This makes the design to be a complex one. But the training data was of 2 years which is not a large number of training data. The loss value was mostly minimized apart from the few spikes time to time in Figure 5. This can be resolved by tuning the hyper-parameters.

If we look at Figure 6, the agent has not been able to gain an increased profit for a longer period of time. Initially, the curve is slightly decreasing. The increase in profit was the highest between 1000 and 1200 iterations. After which the agent forgot the policy and did no learn anything until around 1650 iterations. So, overall, the learning of the agent is not balanced enough to maximize the investment return.

Fig. 6: Profit during Training the Actor-Critic Agent. We have recorded the data each 10 iterations. Fluctuations in the profit curve indicates unstable learning. The agent continued to forget policy after learning for a while.

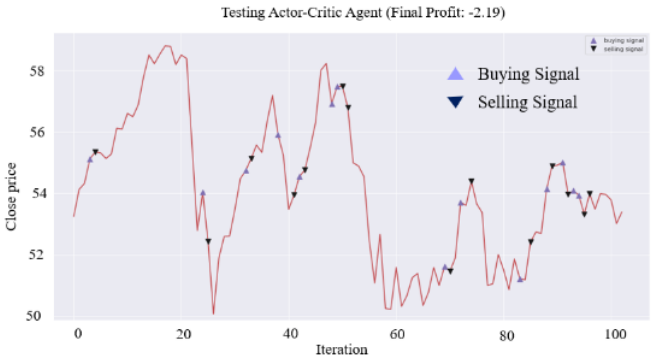

Fig. 7: Testing the Actor-Critic Agent with unseen data of 5-months. After trading in for 5 months, the final profit is -2.19.

Like the deep Q-learning agent, we have also tested the Actor-Critic agent with the 5 months data of trading data. The final profit of this agent was comparatively better than the previous one. Looking at Figure 7, the reason behind the Actor-Critic agent has low consecutive losses could be not doing much trading when the price was higher; which is also true when the price was lower.

VII. Conclusions and Future Work

Being successful in stock market trading is a hard nut to crack because of its instability, unpredictability and nonvolatile nature. The position sizing plays a big role which is defining the size of trading; too much and too little, both will suffer in making a good profit. We have tried to increase our investment return by training two reinforcement learning agents. The agents used two different learning algorithms; one is deep q-learning algorithm and another one is actor-critic reinforcement learning algorithm. The deep q-learning agent used a neural network to estimate the probabilities of taking three actions at every time-step whereas in actor-critic agent, there were two different neural networks to estimate the q-values, evaluating the actions and maximizing the rewards. Among the two agents, the deep q-learning agent did better because the loss was minimized and the profit was maximized iteratively without having any massive increase or decrease respectively. This allows a good learning for the agent. The actor-critic network had significant fluctuations which could have been because of the small size of training data. As the design of the actor-critic network is very complex, a large amount of training data would be good for a stable learning. In our experiments, whenever we were trading, we were buying or selling one unit of stock which could have been modified by adding another logic of determining the size of trading. Also, the long or short position of the stocks could be another good part of the state description of the agents. These modifications could provide a better reference for agents to increase investment returns.

References

[1] Forest Agostinelli, Stephen McAleer, Alexander Shmakov, and Pierre Baldi. Solving the rubik's cube with deep reinforcement learning and search. 1, 2019.

[2] Muhammad Reza Alimoradi and Ali Husseinzadeh Kashan. A league championship algorithm equipped with network structure and backward q-learning for extracting stock trading rules. *Applied soft computing*, 68:478–493, 2018.

[3] Saud Almahdi and Steve Y Yang. An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown. *Expert Systems with Applications*, 87:267–279, 2017.

[4] Shahrokh Asadi, Esmaeil Hadavandi, Farhad Mehmanpazir, and Mohammad Masoud Nakhostin. Hybridization of evolutionary levenberg–marquardt neural networks and data preprocessing for stock market prediction. *Knowledge-Based Systems*, 35:245–258, 2012.

[5] Stelios D Bekiros. Heterogeneous trading strategies with adaptive fuzzy actor–critic reinforcement learning: A behavioral approach. *Journal of Economic Dynamics and Control*, 34(6):1153–1170, 2010.

[6] José Manuel Berutich, Francisco López, Francisco Luna, and David Quintana. Robust technical trading strategies using gp for algorithmic portfolio selection. *Expert Systems with Applications*, 46:307–315, 2016.

[7] Pei-Chann Chang, T Warren Liao, Jyun-Jie Lin, and Chin-Yuan Fan. A dynamic threshold decision system for stock trading signal detection. *Applied Soft Computing*, 11(5):3998–4010, 2011.

[8] Ching-Hsue Cheng, Tai-Liang Chen, and Liang-Ying Wei. A hybrid model based on rough sets theory and genetic algorithms for stock price forecasting. *Information Sciences*, 180(9):1610–1629, 2010.

[9] Ya-Wen Chang Chien and Yen-Liang Chen. Mining associative classification rules with stock trading data–a ga-based method. *Knowledge-Based Systems*, 23(6):605–614, 2010.

[10] Yue Deng, Feng Bao, Youyong Kong, Zhiquan Ren, and Qionghai Dai. Deep direct reinforcement learning for financial signal representation and trading. *IEEE Transactions on Neural Networks and Learning Systems*, 28(3):653–664, 2017.

[11] Yue Deng, Youyong Kong, Feng Bao, and Qionghai Dai. Sparse coding-inspired optimal trading system for hft industry. *IEEE Transactions on Industrial Informatics*, 11(2):467–475, 2015.

[12] Thomas G Fischer. Reinforcement learning in financial markets-a survey. Technical report, FAU Discussion Papers in Economics, 2018.

[13] Jinke Li, Ruonan Rao, and Jun Shi. Learning to trade with deep actor critic methods. In *2018 11th International Symposium on Computational Intelligence and Design (ISCID)*, volume 2, pages 66–71. IEEE, 2018.

[14] Xiao-Yang Liu, Zhuoran Xiong, Shan Zhong, Hongyang Yang, and Anwar Walid. Practical deep reinforcement learning approach for stock trading. *arXiv preprint arXiv:1811.07522*, 2018.

[15] Xiao-Yang Liu, Zhuoran Xiong, Shan Zhong, Hongyang Yang, and Anwar Walid. Practical deep reinforcement learning approach for stock trading. 3, 2022.

[16] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518, feb 2015.

[17] Mahdi Pakdaman Naeini, Hamidreza Taremian, and Homa Baradaran Hashemi. Stock market value prediction using neural networks. In *2010 international conference on computer information systems and industrial management applications (CISIM)*, pages 132–136. IEEE, 2010.

[18] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[19] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[20] Maria Terekhova. Jpmorgan takes ai use to the next level.

[21] Philip Treleaven, Michal Galas, and Vidhi Lalchand. Algorithmic trading review. *Commun. ACM*, 56(11):76–85, nov 2013.

[22] Haozhe Wang, Yulei Wu, Geyong Min, Jie Xu, and Pengcheng Tang. Data-driven dynamic resource scheduling for network slicing: A deep reinforcement learning approach. *Information Sciences*, 498:106–116, 2019.

[23] Yuxin Wu and Yuandong Tian. Training agent for first-person shooter game with actor-critic curriculum learning. 2016.

[24] Zihao Zhang, Stefan Zohren, and Stephen Roberts. Deep reinforcement learning for trading. *The Journal of Financial Data Science*, 2(2):25–40, 2020.