Comparing Model-Free Deep Reinforcement Learning Algorithms on Stock Market

Murat Kaan Meral[1] *
[1]TU Delft

Abstract

Automated asset trading is a crucial method used by financial entities such as investment firms or hedge funds. It allows them to allocate their capital in order to maximize their rate of returns. In scientific literature, there are multiple models suggested to solve this problem. However, these models either lack the complexity to understand the market or scalability for the market in general. On the other hand, deep reinforcement learning is a great framework that can solve these problem. In this study we aim to understand the performance of model-free deep reinforcement learning algorithms in terms of training speed, financial performance and generalizability by training and comparing them on a smaller representative market. Proximal Policy Approximation (PPO) and Twin Delayed Deep Deterministic Policy Gradient (TD3) were used as a representatives of policy approximation and Q-Learning algorithms respectively. Our study have found that while proximal policy algorithms offer higher speed due to smaller training data they use at each timestep, Q-Learning algorithms offer a better general performance in terms of stability and generalizability. With respect to financial performance on training stocks, this study did not find a statistically important difference in performances.

1 Introduction

Automated asset trading is an important part of the strategies of investment companies and hedge funds. It is used to allocate capital and maximize return of investment from the market. However, it is not an easy job for analysts as the asset pricing are stochastic, the market is second order chaotic system and follows a random distribution function, namely geometric brownian motion [5]. Therefore, this makes it harder for them to consider all the relevant factors in this complex system [2; 9].

There are multiple models and solutions developed in order to maximize the return such as optimizing volatility from the capital asset model, or using Markov decision process. However, these solutions either can not capture the whole complexity of the system or too costly to implement on scale [9].

On the other hand, deep reinforcement learning (DRL) is a great strategy for problem solving in unknown dynamic environments. The nature of DRL allows it to have to important advantages: portfolio scalability and market model independence. In finance, stock trading is simply making continuous decision on whether to solve, on what price and when in a complex stochastic market [6].

So far, there have been multiple initiatives in order to combine automated asset trading and deep reinforcement learning. For instance, a library in order to streamline the development of trading algorithms on the stock market has been developed. The library, FinRL, allows researchers to develop new algorithms, and compare them with optimized baselines in multiple use cases such as multiple stock trading and portfolio allocation [6]. Moreover, techniques have been introduced that aims to improve the performance of reinforcement learning algorithms such as data augmentation on finance, and ensemble strategies.

However, literature that focuses on intersection of finance and baseline deep reinforcement learning algorithms is incomplete. For example, Liu et al. [6] shows a table of performances for some algorithms in order to introduce the library, but does not discuss the algorithms' performances in detail. Furthermore, multiple papers also discuss the performance of baseline algorithms, yet does not go into detail on their performance [8; 9; 10]. While these papers, individually, provide us with insights into the performances of these agents, none of them detail the performance of baseline agents in comparison to each other.

This research aims to understand the performance difference of actor-critic based model-free deep reinforcement learning algorithms on the stock market in terms of training speed, performance, and generalizability. In order to do so, two algorithms will be benchmarked: Proximal Policy Optimization (PPO) which optimizes the policy directly [7], and Twin Delayed Deep Deterministic policy gradient algorithm (TD3) which uses a combination of policy optimization and Q-

*Supervised by Greg Neustroev (G.Neustroev@tudelft.nl) and Mathijs de Weerdt (M.M.deWeerdt@tudelft.nl)

Learning [4].

2 Methods

Stock market is a stochastic environment where interactively agents can invest in stocks and get returns based on the change in value of the stocks they own at that given moment. This nature of the environment allows for it to be modeled as a Markov Decision Process (MDP) problem.

Building on top of the Markov Decision Processes, reinforcement learning framework allows artificial agents to trade and improve their performances by interacting with the environment. Each day, agents can take actions to sell or acquire new stocks in order to maximize the rate of return they get from their portfolios. Later, they can observe the environment, and calculate their rewards, which they can use in order to evaluate and optimize their trading strategy. By iterating this process multiple times, the agents can learn more complex strategies and outperform the market average.

This paper formulates the stock environment as a Markov Decision Process, and uses reinforcement learning framework in order to train and evaluate different algorithms. In order to do so, a smaller representation of market is created with only two stocks. The Section 2.1 will be explaining how the market is setup, how stocks were selected, and what information can the agents access. Then, Section 2.2 will go into detail of which algorithms were selected to be tested, how those algorithms were trained, and how they were evaluated.

2.1 Environment

The environment used in this paper is modelled as a Markov Decision Process, in which an agent can observe the state of the environment -which is the stock market-, can take actions -to buy or sell stocks-, and receive a reward -the rate of return from their portfolio-.

Figure 1: A diagram showing the interaction stock traders have with the environment as a Reinforcement Learning problem.

In this subsection, first the stock trading environment will be explained in Subsection 2.1. Then, the selected stocks for training and comparing the algorithms on will be given along with its reasoning in Subsection 2.1.

States, Actions, and Rewards

This paper is using StockTradingEnvCashpenalty environment based on FinRL Library. The library allows us to use historical data in order to create and run OpenAI Gym environment with multiple stock options [6]. Moreover it implements penalties for out of cash positions. In accordance with the library documentation the observation space, action space, and reward function are as follows.

Observation Space: It describes the observation that have been made by the agent in the stock market. In FinRL library, it contains the following information:

- Balance: Amount of cash the agent has at time step t.
- Shares Own: Number of shares the agent owns for each available stock in the environment.
- Closing/Opening/High/Low Prices: They are used to track the changes of stock prices.
- Trading Volume: Total amount of shares traded in a trading slot for each available stock.
- Technical Indicators: Technical financial indicators such as Moving Average Convergence Divergence (MACD), or Relative Strength Index (RSI). These indicators can be used to further the understanding of how market and specific stocks in these markets behave with respect to their history.

Action Space: It is describes all the actions an agent can possibly take for a given state. In the library, action space is represented as a vector of numbers, where each number represent how much stock to buy or sell for each asset.

Reward Function: The reward function in the framework is calculated as:

$$R_t = \frac{(C_t + A_t) - C_0 - \max(0, ((C_t + A_t) * \alpha) - C_t)}{t}$$

where C_t and A_t represent the value of cash and assets the agent has at day t, and alpha is set as a parameter for cash penalty proportion.

Stock Selection

The environment uses two stocks in order to simulate market conditions minimally. Using two stocks allows the agents to avoid drawdown periods as best as they can, while maximizing the profits by making selection between multiple stocks.

In order to create a small representation of market, stocks Ford Motor Company (F) and NVIDIA Corporation (NVDA) were selected. The reason these stocks were selected is the diversification of market. Having a diverse market with low correlation and two stocks with different volatility should allow the agents to learn from a diverse set of data, therefore, it should allow the agents to perform their best during evaluation.

Volatility difference in the study allows for agent to learn more from the same market. As it can be seen from the Table 1, while Ford Motor Company has a low standard deviation in prices, NVIDIA Corporation has extremely high volatility. By having a stock with high volatility such as NVDA and low

Tickers	Average Price	Standard Deviation
F	$8.3512	$2.7579
NVDA	$98.0999	$149.5693

Table 1: Average closing price and standard deviation for stocks Ford and NVIDIA Corporation between 2006 and 2021.

volatility such as F allows for agent to see a higher range of states in the market, meaning it increases the diversity in the training data of the agents. Therefore, the agents can learn from a wide variety of market positions, which should lead to better performance during the evaluation.

Tickers	Correlation
F - NVDA	0.2363
F - DOW	0.8998
NVDA - DOW	0.6455

Table 2: A table displaying the correlations between three tickers F for Ford, NVDA for NVIDIA, and DOW for Dow Jones Index representing the market average.

Another important reason these stocks were chosen was that these stocks are from different sectors. Having stock from different sectors makes sure that the correlation between stocks are low. We can see this from the correlation between tickers F for Ford, NVDA for NVIDIA, and DOW for Dow Jones Industrial Average. Comparing the correlations between tickers which is shown on Table 2, we can see that while each stock has relatively high correlation with the market average which is represented by Dow Jones Index, the correlation between the two selected stocks is relatively low.

Choosing two uncorrelated stock allows the agents' decisions to have more importance. For example if the two stocks were highly correlated, then the agent's decisions would have minimal impact on the return on investment, because the decisions would not matter as long as there is money invested on the stocks. Simply put, it would minimize the environment to almost single stock environment. Therefore two stocks from different sectors with low correlation was chosen to evaluate the agents.

2.2 Agents

This section focuses on which algorithms were selected to be representatives of model-free deep reinforcement learning algorithms as well as how the agents were trained and optimized. First, Subsection 2.2 focuses on what algorithms are selected to be compared as representatives of policy learning and q-learning algorithms, and why. Then, Subsection 2.2 explains how these agents were optimized and trained in learning phase. Finally, Subsection 2.2 describes how the hyperparameters of these algorithms were optimized using distributed learning.

Algorithm Selection

In order to compare the two different types of model free deep reinforcement learning algorithms (namely, policy optimization algorithms, and Q-Learning based algorithms) representatives are chosen. For policy optimization algorithms, PPO was chosen due to its sample efficiency and lower sensitivity to hyperparameters. On the other hand for Q-Learning based algorithms, TD3 was chosen due to the improvement in stability and performance it brings over DDPG.

Proximal Policy Optimization (PPO) algorithm was chosen as the on policy learning algorithm due to its sample efficiency and its decreased sensitivity compared to other policy gradient methods. Normally policy gradient methods are well known for their sensitivity to step sizes; if it is too low, the progress is too slow; and if it is too high, the algorithm can not learn due to noise [7]. On the other hand PPO tries to strike a balance sample efficiency and ease of tuning. Therefore, PPO was selected to be a representative in this study.

Twin Delayed Deep Deterministic Policy Gradient (Twin Delayed DDPG or TD3) algorithm is a stable and high performing value based reinforcement learning algorithm. The algorithm uses actor critic architecture, and builds upon Double Q-Learning in order to avoid problems such as function approximation errors [4]. It increases the stability of the algorithm by using multiple critic networks, and combining their output.

Moreover, TD3 can perform better than DDPG. It can achieve such performance by using additional techniques such as delaying policy updates in order to reduce error per update [4]. TD3 being stable and high performing led to it being selected as representative for Q-Learning methods for this study.

Training and Evaluation

The algorithms were trained on a simulated environment using historical data. For training, the data between 2006 and 2015 was used. The agents were trained using random starting capital sizes with 200,000 timesteps.

For evaluation, backtesting method is used, which assesses how good a trading strategy is [3]. Historical data between 2016 and 2020 is used to simulate the evaluation environment. From the agents' positions over time, multiple financial indicator's are calculated in order to evaluate the performance agents' strategies including cumulative returns and Sharpe ratio.

Hyperparameter Optimization

One of the most important aspects of training and optimizing an algorithm on a given environment is the selection of hyperparameters. The selection can greatly impact the training and the performance of the agent. For example, if learning rate for the agents are too small, the agent can take too long to find a minimum for the neural networks of these agents. On the other hand, big learning rates can cause the neural networks to never converge, which could lead to the agent never learning optimal strategy.

Therefore, Optuna Framework was used in order to optimize hyperparameters for these algorithms. The framework optimizes hyperparameters by running different trials with different combinations of hyperparameters, and then uses the reward (in this case cumulative rate of return) in order to com-

Algorithms (Stocks)	Cumulative Return	Annual Volatility	Sharpe Ratio	Max Drawdown
PPO (F, NVDA)	16.552928	0.468879	1.459257	-0.558039
TD3 (F. NVDA)	16.559846	0.468114	1.461074	-0.557638
PPO (GM, AMD)	31.933818	0.606478	1.451827	-0.485779
TD3 (GM, AMD)	39.035147	0.632949	1.476289	-0.489509
PPO (PG, MCD)	0.864848	0.202265	0.719159	-0.342428
TD3 (PG, MCD)	0.984922	0.222569	0.728766	-0.360291

Table 3: A table showing overview of the performance of each agents over multiple stock combinations. The agents were trained using stocks F and NVDA, and these values were generated using backtests.

pare trials by performance, and suggest better hyperparameters for future trials [1].

Moreover, the agents were were optimized using distributed learning approach. This approach was used in order to parallelize the training among multiple machines, and essentially almost triple the amount of trials that can be run for each algorithm. In total of six Amazon Web Services Elastic Compute Instances were were used in order to optimize the hyperparameters for these algorithms.

3 Results

In this section the results from the study are presented. The results are separated into multiple subsections. The first three subsections will be going into detail of the algorithms' performance during training. Then, Subsection 3.4 will explain how the agents have performed in evaluation stage using backtesting. The algorithms were evaluated on multiple stock combinations including the stocks they were trained with, stocks from the same sectors as the training stocks, and stocks from unrelated sectors.

3.1 Speed

PPO agents on average were faster to train then TD3 agents. The agents were trained on a distributed system that included six Amazon Elastic Compute Cloud (AWS EC2) instances in total, split into two groups of three for each algorithm. For each agent timestep limitation was 200,000. Having the same time limitation, the instances were able to train 35 TD3 agents. On the other hand number of trained PPO agents were 59. This shows as a pure cost of training, PPO is faster to train which allows to iterate more while running hyperparameter optimization with limited compute capabilities.

3.2 Hyperparameter Sensitivity

On-policy deep reinforcement learning algorithms are well known to be highly sensitive to hyperparameters [7]. This is due to the fact that they are only using the data collected with the latest policy in order to train at each timestep. If an agent does not properly explore at the beginning of training or take a single bad step, it can converge to nonoptimal paths, and optimize to a low performing local optimum instead. Therefore, the hyperparameters set at the beginning of the training has huge impact on agents' performances.

PPO is no exception to this rule of hypersensitivity to hyperparameters. Even though PPO is an improvement on on-policy algorithms [7], it still is sensitive compared to Q-Learning algorithms such as TD3. This can be seen from

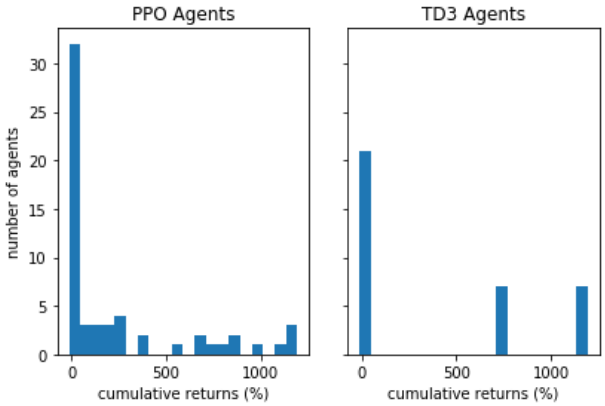


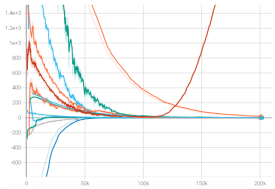
Figure 2: A histogram displaying number of trials against the rate of return. Rate of returns are shown on X axis. The number of agents that corresponds to each rate of return from this study is displayed on the Y axis.

Figure 2, TD3 agents can be seen as rather stable, as they converge into three local optima. On the other hand PPO agents are scattered across, and most of them perform rather poorly. This shows that TD3 algorithm is less sensitive to changes in hyperparameters therefore more stable.

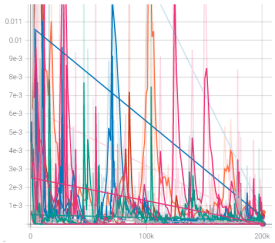
3.3 Frame Efficiency

TD3 can learn more per timestep then PPO. This is due to TD3 being an off-policy algorithm, since it can learn more at each timestep compared to PPO. In off-policy algorithms, the agent can utilize the frames that were seen previously. Accordingly at each timestep, TD3 can learn from all of its history, while PPO can only learn from the latest data.

From Tensorboard outputs of losses for each agent from Figure 3, it can be seen that most of the TD3 agents have converged and learned by timestep 100,000. On the other hand most of the PPO agents do not converge into minimum loss directly, and rather make huge jumps from timestep to timestep. This is a direct result of how PPO learns, and how too large or bad steps can affect the performance of the agents. As a consequence of these bad steps, the samples collected on the latest policy can be useless, which further decreases the frame efficiency of PPO.



(a) TD3 Actor Loss



(b) PPO Loss

Figure 3: Two figures displaying the losses of algorithms during training. First figure displays the actor loss from TD3 agents. Second figure displays the loss of PPO agents.

3.4 Performance

Training Stocks

On the stocks the agents have trained on (Ford Motor Company and NVIDIA Corporation) the best performing agents from each algorithm have performed equally. There were no statistically important differences, as it can be seen from the Table 3 which shows the cumulative returns and annual volatility of each agents' strategies. Moreover, both agents were able to outperform market average by 43% based on their rate of returns.

Same Sector Stocks

When evaluated against different stocks from the same sectors -automobile and chip making-, TD3 was able to perform better than PPO. The agents were evaluated on General Motors-AMD and Volkswagen-Intel pairs with the same parameters as their training. In either of the pairs, best performing TD3 agent were able to outperform best performing PPO agent.

As it can be seen from Figure 4, TD3 agent was able to outperform PPO agent by 21% based on their rate of returns. The better performance is achieved by acquiring more volatile stocks, as TD3 agent's daily volatility average is 4% higher than PPO agent's, yet the Sharpe Ratio is still higher. This means that TD3 agent was taking more risk that still has higher expected returns.

Different Sector Stocks

When compared on the unrelated stock pairs such as Procter & Gamble Co and McDonald's, neither of the agents were able to perform better than the average stock price. TD3 agents perform 8% less than average stock price, while the PPO agent performs 19% worse. As it was the case for same sector stocks, it can be seen that TD3 agent has a slightly higher Sharpe Ratio and slightly higher annual volatility, which translates into the difference in the cumulative rate of return.

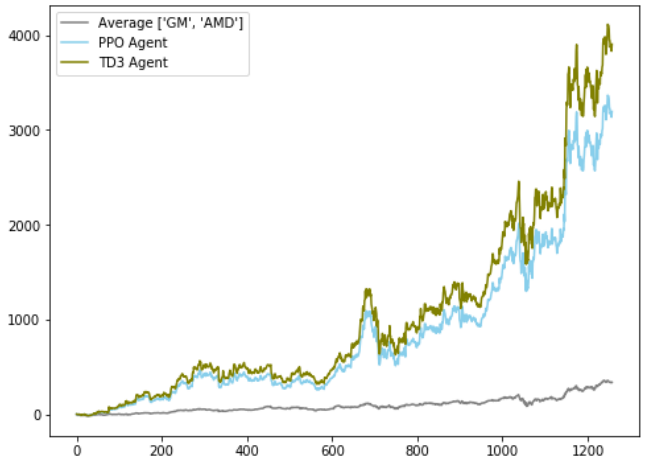


Figure 4: A line graph displaying the rate of return on Y axis and time since the start of trading, 1st of January 2016, on the X axis. This graph shows that the performance of TD3 agent is 21% higher than PPO agent on different stocks from same sector.

4 Reproducibility of Study

Jupyter notebooks used in this study can be find in Delft University of Technology's GitLab server [1]. The codebase includes necessary notebooks in order to train agents, evaluate them, and graph their performances. There were no preset random seeds for neural network generation or hyperparameter optimization sampling. Therefore results generated with the same code can differ. However, given enough trials the results should be similar as law of large numbers applies.

5 Discussion

This section expands on the results provided in Section 3. It explains how these algorithms work, and which aspects of these algorithms could have lead to the results given. First, it argues that for speed against frame efficiency trade off in Subsection 5.1, PPO is a better option due to how environment is set up. Then, it suggests a possible explanation for the similar performances in evaluation as well as focusing on the stability of the performance in Subsection 5.2. Lastly, it argues on what could be the reason for difference in performance in other markets in Subsection 5.3.

5.1 Speed and Frame Efficiency

PPO algorithm is much faster at learning compared TD3 as it was shown in Section 3.1. This is due to the difference in how much data they use at each timestep. Since TD3 is an off-policy algorithm, it uses all of the history to learn at each timestep, while PPO uses only the data generated with latest policy. The difference in size of the data algorithms can train with at each timestep directly affects the training times, because neural networks run a forward propagation and backward propagation for each sample data.

[1] https://gitlab.ewi.tudelft.nl/cse3000/2020-2021/rp-group-34/rp-group-34-mmeral

On the other hand, the disadvantage resulting from training speed can be seen as a trade off between training speed and frame efficiency. Using historical data along with the latest samples, TD3 algorithm can learn more from each sample.

For stock market environments, having more speed rather than frame efficiency is a better choice. The cause of this is the complexity of the environment. Since the environment uses historical data, complexity of each timestep is $O(1)$, meaning the calculation of each frame is cheap. As a result, being frame efficient is less advantageous than being faster per frame.

5.2 Performance

Since there was no statistical difference in performances of best performing agents from each algorithm, we can not meaningfully discuss their financial performance. We believe the reason for the lack of difference in performance is overfitting. For example on losses from TD3 agents from Figure 3, the actor loss converges close to zero around 90,000th step. Since early stopping strategy was not used for the agents, the agents start to overfit to the data. Therefore the agents learn the strategy that leads to best performance in historical data between 2006 and 2015.

In terms of stability, however, TD3 algorithm is performing much better compared to PPO. As discussed in Section 3.2 and can be seen from Figure 2, the average return for TD3 agents is much higher than PPO agents which tend to have a performance collapse. This is caused by sensitivity to hyperparameter, and is a big advantage off-policy algorithms such as TD3, SAC over on-policy algorithms.

5.3 Generalization and Transferability

Q-Learning based agents are more generalizable than agents based on policy optimization algorithms. It can be seen from the difference in performance in same sector stocks, and partially in stocks from different sectors. Q-Learning tries to optimize expected reward ,and uses all the data to learn. Therefore, it can generate models that are more informed and give more universal results.

As a consequence of generalizability of Q-Learning, the models generated can be readily used for similar applications (with different stocks), or require less timesteps to optimize than models generated with policy optimization algorithms. The reward prediction networks can transfer their ability to predict other markets relatively easily, since they are trained with all of the data collected during previous training.

On the other hand, the disadvantage policy optimization is that they are directly optimizing for the actions, which may not be transferable to environments with different stocks. Thus while learning the unknown market, it can stay in local optimum or take bad steps that were optimal for previous markets, which then could lead to performance collapse.

6 Conclusions

This research aimed to understand the performance of actor-critic based model-free deep reinforcement learning algorithms on the stock market in terms of training speed, performance, and generalizability. It did so by comparing two representative algorithms: Proximal Policy Approximation (PPO), and Twin Delayed Deep Deterministic Gradient (TD3). This research has showed that while PPO has a much higher training speed, TD3 was a better algorithm for general performance and ability to transfer learning to other markets.

In terms of training speed, on-policy algorithms are faster than off-policy algorithms. This can be seen from the results, in which the number of PPO agents trained was much higher compared to number of TD3 agents trained. These results can be explained by the amount of data algorithms have to learn at each timestep. While PPO trains on only the latest samples, TD3 trains on all of the recorded samples.

Compared on the stocks they have trained with, both of the algorithms have showed a similar performance. We suspect this is due to overfitting on the training data. Therefore, in the limits of this study, there were no further discussions on the performance of agents on evaluation data. However with the analysis of the optimization process, it can be concluded that off-policy algorithms are more stable than on-policy algorithms. In our study, while close to half of TD3 agents were able to learn and perform relatively well, most of the PPO agents have performed poorly due to performance collapses which are caused by either bad or too big steps in policy change during training.

For generalizing to other markets and transferring of learning, Q-Learning algorithms show themselves to be better compared to policy approximation algorithms. This study shows that when evaluated on different stocks than they were trained with, TD3 algorithm performs better. This can be explained from both of the algorithms' perspectives. First, PPO performs poorly, because it tries to optimize directly on the goal. It focuses on the best actions in situations rather than learning more about how market moves. On the other hand, TD3 does the opposite. It tries to understand and predict the market, which leads to better performance for different markets.

7 Future Work

While this study has shown results with respect to its aim of understanding the performance of actor-critic based model-free deep reinforcement learning algorithms in stock markets, due to the limitations in time and compute capabilities, it was not able to extend to different scenarios where these algorithms could be used. In order to understand these algorithms better, we are proposing the below actions to be included in future research.

First, we believe the agents were overfitting to the training data. Therefore in future applications or research, early stop strategies are crucial in order to optimize the agent. While overfitting decreases the loss during training time and result in better models for training data, it decreases the performance in evaluation sets as well as other stocks.

We believe future studies requires scaling in order to validate the findings of this research on bigger markets as well as different combinations. Increasing the market size from two will allow agents to have a bigger action space, therefore increasing any performance difference the algorithms might have in the market.

Additionally, the algorithms can be trained on multiple markets simultaneously. This would allow the agents to learn more about how market moves rather than focusing on the movement of each specific stock. Using such strategy should allow the generalizability to be evaluated more easily.

References

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[2] Stelios D. Bekiros. Fuzzy adaptive decision-making for boundedly rational traders in speculative stock markets. *European Journal of Operational Research*, 202(1):285–293, 2010.

[3] James Chen. Backtesting. https://www.investopedia.com/terms/b/backtesting.asp/, 2021. [Online; accessed 2021-06-26].

[4] Scott Fujimoto, Herke Van Hoof, and David Meger. Addressing Function Approximation Error in Actor-Critic Methods. *35th International Conference on Machine Learning, ICML 2018*, 4:2587–2601, 2018.

[5] John C. Hull. *Risk management and financial institutions*. Wiley, 5th edition, 2018.

[6] Xiao Yang Liu, Hongyang Yang, Qian Chen, Runjia Zhang, Liuqing Yang, Bowen Xiao, and Christina Dan Wang. FinRL: A deep reinforcement learning library for automated stock trading in quantitative finance. *arXiv*, (NeurIPS):1–11, 2020.

[7] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv*, pages 1–12, 2017.

[8] Zhuoran Xiong, Xiao Yang Liu, Shan Zhong, Hongyang Yang, and Anwar Walid. Practical deep reinforcement learning approach for stock trading. *arXiv*, (i), 2018.

[9] Hongyang Yang, Xiao-Yang Liu, Shan Zhong, and Anwar Walid. Deep Reinforcement Learning for Automated Stock Trading: An Ensemble Strategy. *SSRN Electronic Journal*, 2020.

[10] Yuyu Yuan, Wen Wen, and Jincui Yang. Using data augmentation based reinforcement learning for daily stock trading. *Electronics (Switzerland)*, 9(9):1–13, 2020.