Department of Energy Engineering, Sharif University of Technology
Term paper of the Reinforcement Learning course- Autumn 2023

Deep Reinforcement Learning Agents in Financial Markets

Pooriya Safaei[1], **Maryam Asgarinezhad**[2],
Prof. Mehrdad Boroushaki[3]

[1]Department of Mathematics and Computer Science, Sharif University of Technology, Tehran; pooriya.safaei@sharif.edu
[2]Department of Mathematics Computer Science, Sharif University of Technology; maryam.asgarinezhad@sharif.edu
[3]Department of Energy Engineering, Sharif University of Technology, Tehran; boroushaki@sharif.edu

Abstract

The field of Quantitative Trading (QT) is experiencing a significant transformation, thanks to the emergence of cutting-edge technologies such as Artificial Intelligence (AI) and Machine Learning (ML). These advanced algorithms have the capability to analyze large volumes of financial data and identify profitable investment opportunities with unmatched speed and accuracy, surpassing the abilities of human traders. The automation of critical tasks such as market analysis and trading execution has been a game-changer in the QT domain. However, the current QT methods face a few persistent challenges, particularly in handling noisy and high-frequency financial data. Additionally, maintaining a balance between exploration and exploitation is a significant challenge for AI-driven trading agents. To overcome these obstacles, we propose a solution called QT-Net, which introduces an adaptive trading model that autonomously creates QT strategies through an intelligent trading agent. Our model employs deep reinforcement learning (DRL) with imitative learning methodologies to enhance its proficiency. To tackle the challenges posed by volatile financial datasets, we conceptualize the QT mechanism within the framework of a Partially Observable Markov Decision Process (POMDP). Moreover, by embedding imitative learning, the model can capitalize on traditional trading tactics, nurturing a balanced synergy between discovery and utilization. To simulate real-world conditions, we train our trading agent using minute-frequency data sourced from live financial markets. Our experimental findings highlight the model's ability to extract robust market features and its adaptability to diverse market conditions.

Keywords: Deep Reinforcement Learning, Financial Market Analysis, Risk Management, Quantitative Trading

Introduction

In the domain of financial security investment, quantitative trading (QT) stands out for its extensive automation, leveraging computing technology to reduce reliance on personal discretion and minimize irrational decision-making. With the escalating prominence of quantitative hedge funds, there is a growing emphasis on incorporating machine learning into QT, particularly within the realm of Fintech. These technologies facilitate the development of dynamic trading strategies capable of adapting to real-time market changes, effectively managing risks, and ultimately bolstering the profitability and efficiency of trading operations. In the face of an increasingly intricate financial landscape, the integration of artificial intelligence (AI) and machine learning (ML) in QT has become indispensable for maintaining a competitive edge.

Within the volatile environment of financial markets, trading behaviors and economic events are inherently unpredictable, leading to the generation of volatile and non-stationary data. Despite the widespread use of technical analysis in Quantitative Trading (QT), its capacity for generalization has been questioned, highlighting the urgent need for more robust features directly mined from raw financial data. In response, machine learning techniques, particularly deep learning models, have been explored for their potential to predict market trends and enhance generalization. However, the scope of QT extends beyond trend prediction; it necessitates the formulation of strategic trading methods.

Reinforcement learning (RL) provides a systematic framework for addressing tasks involving a sequence of decisions, yet striking a balance between discovering novel strategies and utilizing established ones presents a significant challenge, especially within the pragmatic constraints of actual trading scenarios. In response to these challenges, we introduce Observational and Recurrent Deterministic Policy Gradients (QTNet). We position QT within the framework of a Partially Observable Markov Decision Process (POMDP) to effectively handle the representation of unpredictable financial data. Recurrent neural networks, such as Recurrent Deterministic Policy Gradient (RDPG), are employed to manage the POMDP, offering an off-policy deep reinforcement learning (DRL) algorithm.

The challenge of balancing exploration and exploitation is addressed through imitative learning techniques. A demonstration buffer, initialized with actions from Dual Thrust, and behavior cloning are introduced to guide the trading agent. By integrating these techniques into the POMDP framework, QTNet leverages enhanced financial domain knowledge. Real financial data from the futures market is employed to test QTNet, showcasing its

ability to learn profitable trading policies and exhibit superior generalization across various futures markets.

Elements of a Paper

Here are what we are going to see in this paper:

- Problem Definition with RL
- Foundation
- QT as a POMDP
- DRL Agents used in this paper
- PPO performance on single symbol trading
- Performance of multiple stock trading
- Demonstration and Conclusion
- Possible Future Works

Problem Definition with RL

In this section, precise definitions for mathematical symbols are provided, followed by a thorough formal introduction to elaborate on the complexities of the quantitative trading problem.

Foundation

For each discrete moment t, we define the OHLC price array as $p_t = [o_t^p,\ h_t^p,\ l_t^p, c_t^p]$ correspondingly represent the opening, highest, lowest, and closing prices. The comprehensive price array for a financial instrument is expressed as $P = [p_1, \dots, p_t]$. The technical indicator vector at time t is denoted as $I_t = \left[I_{1t}, \dots, I_{jt}\right]$ where I_{kt} is a function of the historical price sequence from timestep 1 to t. Similarly, the account profit at time step t is r_t and the sequence of it is $R = [r_1, \dots, r_t]$. Note that we cannot use only the price as the environment for the RL problem. This is because the state space would be extremely large, and we would lose recurrency. It's important to keep in mind that using the price alone as the environment for a reinforcement learning (RL) problem is not always feasible. This is because the state space can become extremely large and we risk losing the ability to capture the recurrent patterns that are essential for successful RL. To overcome this challenge, we can use indicators that are derived from the time series of prices. These indicators serve as a function of the prices and normalize the environment within a certain range, making it easier for the agent to learn and adapt. By leveraging such techniques, we can create a more effective and efficient RL system. Here in this paper we are using some simple indicators that are common in financial markets such as RSI, MACD, BB, CCI, and etc.

POMDP

In this section, we will delve into the unique features of Quantitative Trading (QT) and explain why it is advantageous to view the entire QT process as a Partially Observable Markov Decision Process (POMDP). In the financial market, the prices of securities are influenced by numerous factors, such as bullish and bearish investors, macroeconomic and microeconomic activities, unforeseeable events, and varied trading behaviors, resulting in inherent noise. Due to these complexities, direct observation of the true market states becomes impractical. For instance, the effect of positive news or order executions is uncertain, and the data available for analysis is limited to historical prices and trading volumes, providing only a partial picture of the market state. QT, as a sequential decision-making challenge, revolves around determining what and when to trade.

Within the realm of Quantitative Trading QT, the analytical structure evolves from a classic Markov Decision Process (MDP) to a Partially Observable Markov Decision Process (POMDP). An MDP is defined by a quintuple [S, A, P, R, γ], where S denotes a distinct set of states, A represents a specific set of actions, P is the state transition probability function, and R signifies the reward function. By incorporating observational elements O and Z, with O being the observation set and Z the observation probability function, the framework is modified into a POMDP. In this context, the agent receives various observations at each interval, and the comprehension of observable history until time t is fundamental in capturing the dynamics of QT.

Observation. When it comes to the financial market, it's important to note that observations can be classified into two main categories - the portfolio observation subset (o_{pt}) and the economic observation subset (o_{et}). The o_{pt} subset refers to the total profit of the portfolio, while the o_{et} subset is associated with market pricing and technical metrics. To get a comprehensive overview, it's essential to consider both subsets, which together make up the overall observation set O. This set, denoted as O = P, I, R, also includes technical indicators such as BuyLine and SellLine from the Dual Thrust strategy. By analyzing all these factors, investors can make informed decisions and develop effective trading strategies.

Action. trading actions involve a continuous probability vector at representing long and short positions. The agent executes the action associated with the maximum probability. Actions are represented as $a_t \in$ { long , short, neutral } = {1, −1, 0}, simplifying position management and mitigating the impact of market capacity. “long” means agent must buy here, “short” means the flag of sell for the agent, and “neutral” means that the agent must not act and only has to watch the market and waits for the next state. Trading actions are interpreted as signals, guiding the execution of trades based on specific rules.

Reward. In order to simulate trading more accurately, practical considerations such as transaction fees and

slippage are taken into account. This means that the account profit, represented as "r_t", is calculated based on these factors. However, it is important to note that r_t is not always the most effective way to evaluate performance. For this reason, the Sharpe ratio is used as a risk-adjusted measure of return. The Sharpe ratio, or "S_r", is a metric that takes into account the excess returns compared to the total risk of an investment.

This framework provides a thorough and comprehensive approach to simulating trading in a dynamic and uncertain financial environment. By incorporating realistic market constraints, it is capable of representing the challenges faced by quantitative traders.

DRL Agents

Our next focus is on presenting some sophisticated Deep Reinforcement Learning (DRL) agents that we plan to employ in our paper. To facilitate the training of these models and agents, we have opted for the FinRL library, which offers a rich repository of data from various sources and markets like FOREX, NASDAQ, and more. This powerful tool will enable us to harness the potential of DRL and analyze the performance of our agents in real-world scenarios.

PPO:

Proximal Policy Optimization (PPO) is a powerful deep reinforcement learning (DRL) algorithm designed to optimize policies for sequential decision-making tasks. PPO belongs to the family of policy gradient methods and is known for its stability and sample efficiency. Unlike some earlier algorithms, PPO seeks to prevent large policy updates that might disrupt training by introducing a clipped surrogate objective function. This clipping mechanism constrains the change in the policy during each iteration, ensuring more stable and controlled learning. PPO has demonstrated success in various applications, including financial trading, where its ability to handle continuous action spaces and efficiently optimize policies makes it well-suited for adapting to the dynamic nature of market conditions. The algorithm's balance between stability and performance has contributed to its popularity and widespread use in diverse reinforcement learning scenarios.

A2C:

Advantage Actor-Critic (A2C) is a deep reinforcement learning (DRL) algorithm that combines elements of both policy gradient and value function methods. A2C aims to address some of the challenges faced by traditional policy gradient methods, such as high variance in gradient estimates, by introducing an advantage function. This function represents the advantage of taking a specific action in a given state over the average action value. The actor network is responsible for generating a policy, mapping states to actions, while the critic network estimates the state-value function. A2C employs parallelization to improve sample efficiency, allowing multiple agents to explore the environment simultaneously. This algorithm has shown effectiveness in various applications, including financial trading, where it excels in learning complex and adaptive strategies to navigate dynamic market conditions.

SAC:

Soft Actor-Critic (SAC) is a state-of-the-art deep reinforcement learning (DRL) algorithm designed for continuous action spaces, making it well-suited for applications in financial trading where actions are often nuanced and continuous. SAC combines key features of policy optimization and value-based methods, employing an entropy-regularized actor-critic framework. The introduction of entropy regularization encourages exploration and enhances the robustness of learned policies. SAC simultaneously learns both the policy and the Q-function, allowing for efficient optimization in high-dimensional state and action spaces. The algorithm prioritizes both maximizing expected cumulative rewards and maximizing policy entropy, striking a balance between exploration and exploitation. SAC has demonstrated success in various domains, showcasing its adaptability and effectiveness in handling the challenges posed by dynamic and uncertain financial markets.

TD3:

Twin Delayed DDPG (TD3) is an advanced deep reinforcement learning (DRL) algorithm that builds upon the strengths of Deep Deterministic Policy Gradient (DDPG) while addressing some of its limitations. TD3 introduces several key innovations to enhance stability and sample efficiency in learning complex policies for continuous control tasks, such as those found in financial trading. Notably, TD3 utilizes twin critics, incorporating two separate critic networks to estimate the value of actions. This mitigates overestimation bias and stabilizes the learning process. Additionally, TD3 introduces a target policy smoothing technique, which adds noise to the target action during the learning process to improve robustness. These modifications make TD3 particularly well-suited for applications in financial markets, where the ability to handle continuous action spaces and maintain stability during training is crucial for effective policy optimization.

DDPG:

Deep Deterministic Policy Gradient (DDPG) is a powerful deep reinforcement learning (DRL) algorithm designed for continuous action spaces, making it

particularly suitable for financial trading scenarios where actions are often nuanced and continuous. DDPG combines elements of both policy-based and value-based methods, utilizing an actor-critic architecture. The actor network determines the optimal policy, mapping states to actions, while the critic network evaluates the policy by estimating the expected cumulative reward. Notably, DDPG employs target networks to stabilize learning, mitigating issues related to policy oscillations. This algorithm excels in handling high-dimensional state and action spaces, providing a robust framework for optimizing complex trading strategies in the dynamic and uncertain environment of financial markets.

PPO performance on single symbol trading

To prepare for agent training, we must first introduce two essential constraints into the environment and action space. These constraints serve to abstract the trading process and ensure that it is as realistic as possible. One such constraint is the inclusion of transaction costs, which are incurred whenever a trade is executed. Transaction costs can take many forms, such as broker commissions and SEC fees. By including these costs as a constraint, we can better simulate real-world trading scenarios and ensure that our agents are prepared to handle them effectively. These two constraints are:

- *Flat fee:* a fixed money amount per trade regardless of how many shares traded.
- *Per share percentage:* a per share rate for every share traded, for example, 1/1000 or 2/1000, are the most commonly used transaction cost rate for each trade.

Moreover, we need to consider market liquidity for stock trading, such as bid-ask spread. Bid-ask spread is the difference between the prices quoted for an immediate selling action and an immediate buying action for stocks. In our environments, we can add the bid-ask spread as a parameter to the stock closing price to simulate real world trading experience.

We will test the result of the PPO algorithm on a single-symbol portfolio from January 2019 to September 2020 while assuming these constraints. The symbols chosen for the test include Amazon (AMZN), Google (GOOGL), Apple Inc. (AAPL), and others.

Figure 1. Performance of single stock trading using PPO

Looking at the results, it is evident that the PPO algorithm outperformed the AAPL symbol. However, in February 2020, there was a noticeable decline in the performance of the agents. One possible explanation for this could be the environment that was introduced to the agents. In this environment, we incorporated certain Indicators to simplify the state space, which might have resulted in less information available to the trained model. This could have led to errors in the model's predictions, causing a decline in agent performance. But the most significant evidence for this decline, is that the end of 2019 and the beginning of 2020 the COVID-19 pandemic started, and all financial markets and stock experienced a dramatic decline. Because of that for the next part, we use a more updated data from the middle of 2020 till 2023.

2019/01/01-2020/09/23	SPY	QQQ	GOOGL	AMZN	AAPL	MSFT	S&P 500
Initial value	100,000	100,000	100,000	100,000	100,000	100,000	100,000
Final value	127,044	163,647	174,825	192,031	173,063	172,797	133,402
Annualized return	14.89%	32.33%	37.40%	44.94%	36.88%	36.49%	17.81%
Annualized Std	9.63%	27.51%	33.41%	29.62%	25.84%	33.41%	27.00%
Sharpe ratio	1.49	1.16	1.12	1.40	1.35	1.10	0.74
Max drawdown	20.93%	28.26%	27.76%	21.13%	22.47%	28.11%	33.92%

Table 1. Performance of single stock trading using PPO. The Sharpe ratio of all the ETFs and stocks outperform the market, namely the S&P 500 index.

But our primary objective is to assess the effectiveness of various algorithms on a portfolio comprising multiple symbols. We aim to observe the outcomes in this particular scenario and gauge how well these algorithms perform in a real-world situation.

Performance of multiple stock trading

In order to accurately evaluate the performance of RL (Reinforcement Learning) models, it is important to establish a benchmark. For this purpose, we employ the "Mean-Variance Strategy," which is a highly sophisticated and advanced approach commonly utilized in Quantitative Trading. This strategy is recognized for its strength in analyzing and determining optimal investment decisions by taking into account the mean and variance of returns. By

utilizing this approach, we can ensure that our assessments of RL models are comprehensive and reliable.

Mean-Variance Strategy. The Mean-Variance Strategy, rooted in modern portfolio theory, is a fundamental approach in financial investment that aims to optimize the trade-off between expected returns and portfolio risk. Developed by Harry Markowitz, the strategy considers the mean (average) return of an investment and its variance (or standard deviation), assuming that investors are risk-averse and seek to maximize returns while minimizing overall portfolio risk. The strategy involves constructing a diversified portfolio by allocating weights to different assets based on their expected returns and historical volatilities. The goal is to achieve an efficient frontier, representing portfolios that offer the highest expected returns for a given level of risk. Investors can then choose a portfolio along the efficient frontier that aligns with their risk tolerance. The Mean-Variance Strategy has been a cornerstone in investment theory, providing a systematic approach for constructing well-balanced portfolios that optimize risk-adjusted returns.

To train the agents, we need to create a portfolio of specific symbols. The chosen symbols are:

- Apple Inc (AAPL)
- Microsoft Corp (MSFT)
- Meta Platforms Inc (META)
- IBM Common Stock (IBM)
- Home Depot Inc (HD)
- Caterpillar Inc (CAT)
- Amazon.com Inc (AMZN)
- Intel Corp (INTC)
- AT&T Inc (T)
- Visa Inc (V)
- Goldman Sachs Group Inc (GS)

To begin the training process, we first need to set up the environment. Our initial capital for this training exercise will be set at 1,000,000 units. The training period will span from '2009-01-01' to '2020-07-01', while the test period will begin on '2020-07-01' and end on '2023-01-01'. It is important to note that during the train period, the agents will not have access to the test data, and their training will come to a halt.

You can see the result of test on A2C, PPO, SAC in order to compare with the Mean-Variance Strategy.

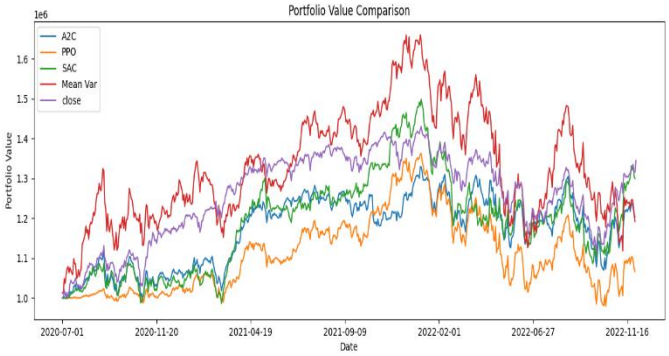

Figure 2. Performance A2C, PPO, SAC stock trading and portfolio allocation

At the end of the given period, it is evident that Agents such as A2C have exhibited similar performance to the mean-variance strategy. However, over time, A2C has started to outperform the mean-variance strategy. On the other hand, the SAC strategy has proven to be the most effective among all RL strategies, surpassing both A2C and the mean-variance strategy. At the end of the given period, SAC has demonstrated a significantly better performance than the other strategies and has exceeded the mean-variance strategy in terms of effectiveness.

There are two important agents whose results have not yet been displayed, DDPG and TD3. I would like to inform you that the results of these agents are crucial and may impact the final outcome significantly. Therefore, it is essential to keep an eye on their results. Please find below the names of these agents whose results are yet to be shown:

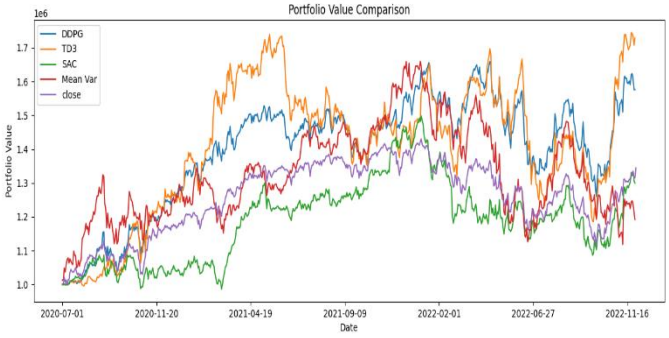

Figure 2. Performance A2C, PPO, SAC stock trading and portfolio allocation

According to recent research, it has been observed that the Deep Deterministic Policy Gradient (DDPG) and Twin Delayed Deep Deterministic Policy Gradient (TD3) agents exhibit the best performance in a variety of reinforcement learning tasks. In fact, after some iterations, both DDPG and TD3 agents outperform the traditional mean-variance strategy. This suggests that these two techniques are more effective in solving complex problems and have the potential to be used in various real-world applications.

	TD3	DDPG	Min-Var.	DJIA
Initial value	1,000,000	1,000,000	1,000,000	1,000,000
Final value	1,403,337; 1,381,120	1,396,607; 1,281,120	1,171,120	1,185,260
Annualized return	21.40%; 17.61%	20.34%; 15.81%	8.38%	10.61%
Annualized Std	14.60%; 17.01%	15.89%; 16.60%	26.21%	28.63%
Sharpe ratio	1.38; 1.03	1.28; 0.98	0.44	0.48
Max drawdown	11.52% 12.78%	13.72%; 13.68%	34.34%	37.01%

Table 2. Performance of multiple stock trading and portfolio allocation over the DJIA constituents Stock. The Sharpe ratios of TD3 and DDPG excceed the DJIA index, and the traditional min-variance portfolio allocation strategy.

Demonstration and Conclusion

Our aim is to showcase the effectiveness of our approach in two specific use cases: single stock trading and multiple stock trading. In order to establish the efficacy of our approach, we have selected these two use cases and produced results using an established benchmark - the Mean-Variance Strategy. This strategy is widely used in the field of quantitative trading and serves as a reliable and established benchmark for comparison purposes. Our results demonstrate the superiority of RL approach in both use cases and highlight its potential for generating superior returns in the stock market.

In our study, we have conducted a performance evaluation of single stock trading, and the results are presented in Fig. 1 and Table 1. For this purpose, we have selected large-cap ETFs such as SPDR S&P Trust (SPY) and Invesco QQQ Trust Series 1 (QQQ), and popular stocks including Google (GOOGL), Amazon (AMZN), Apple (AAPL), and Microsoft (MSFT). To train a trading agent, we have utilized the PPO algorithm in FinRL. We have observed that the maximum drawdown in Table 1 is relatively large, mainly due to the Covid-19 market crash that occurred during the study period. However, despite the temporary market volatility, our analysis demonstrates the potential of utilizing trading agents powered by machine learning algorithms like PPO to achieve better trading outcomes in the stock market.

In our analysis of the performance of various algorithms and agents for multiple stock trading and portfolio allocation over the Dow Jones 30 constitutes, we have presented our findings through Fig. 2 and Table 2. Our study involved the use of five different algorithms, namely PPO, A2C, SAC, DDPG, and TD3. After extensive testing and observation, we have found that the last two algorithms, i.e. DDPG and TD3, have outperformed the other algorithms and agents in terms of overall performance. These findings are significant for traders and investors looking to optimize their trading strategies in the stock market.

However, this study underscores the importance of acknowledging the multifaceted nature of algorithmic trading, where success is contingent on numerous factors beyond algorithm performance alone. Considerations such as data quality, model interpretability, real-world implementation challenges, and ethical implications demand careful attention. Furthermore, the diversity of financial instruments and market conditions introduces additional layers of complexity that necessitate a nuanced approach in algorithm selection and deployment. While DDPG and TD3 have demonstrated exceptional capabilities, the evolving nature of financial markets underscores the ongoing need for research and innovation to address emerging challenges and refine algorithmic trading strategies. This study serves as a stepping stone in understanding the strengths and limitations of DRL algorithms in financial trading, providing a foundation for future developments in this dynamic and rapidly evolving field.

Possible Future Works

The field of reinforcement learning is always evolving, and there are many new ideas that researchers can use to improve its performance even further. One thing to keep in mind is that in real-world problems involving reinforcement learning, the definition of the environment for the agent is largely dependent on the researcher. This means that researchers have a lot of flexibility in defining the environment, and can use a wide variety of inputs to train the agent.

In this context, it's important to note that we used a very simple environment for our experiment, with only basic indicators that are commonly used in the market. However, there are many other inputs that can be used to train the agent, including any set that can be defined as a time series. This opens up a lot of creative possibilities for researchers.

For example, one interesting idea is to give the output of an already-trained agent as an input to another agent. The output of an agent is in the form of a 3, which represents the action that the agent took in a certain state. Thus, the other agent can interpret this action as part of its own environment, and use it to detect and filter any errors made by the first agent. This approach of combining two or more agents in this way can be a powerful way to make progress in the field of reinforcement learning.

Moreover in our experiment, we have trained an agent for a certain period of time and then tested it on a different period of time. However, we observed that as the gap between the training and testing data in the time series increases, any model used to predict the given training loses its overall performance quality. Therefore, it is important not to halt the training of the agent due to the nature of the time series. In other words, we should consider only the present time as a test and, at a moment after the present time, we should convert the present time data into training data and learn from the performed action. This approach helps us eliminate the gap between training and testing and can significantly improve the performance of the agent in the long run. In summary, by continuously training the agent with new data, we can ensure that it adapts to the changing nature of the time series and maintains its predictive accuracy over time.

References

[1] Quang-Vinh Dang. Reinforcement learning in stock trading. In ICCSAMA, 2019.

[2] Andrew Ang. Mean-variance investing. Columbia Business School Research Paper No. 12/49., August 10, 2012.

[3] Stephen Dankwa and Wenfeng Zheng. Twin-delayed DDPG: A deep reinforcement learning technique to model a continuous movement of an intelligent robot agent. Proceedings of the 3rd International Conference on Vision, Image and Signal Processing, 2019.

[4] Nicolas Heess, Jonathan J Hunt, Timothy P Lillicrap, and David Silver. Memory-based control with recurrent neural networks.