Berkeley
UNIVERSITY OF CALIFORNIA

*Capstone Project Report*

# Using Reinforcement Learning and Multi-Agent Simulator to Improve the Profitability and Efficiency of High-Frequency Trading


FinTech Team D2

Team Member:

Charlotte Jin,

Ethan Choukroun,

Jingqi Ma,

Haoyang Wang


April, 2023

# *Report Outline*

# 1. Executive Summary

Reinforcement learning is a machine learning method that finds strategies to maximize the reward. Dealing with millions of data and training itself without humans in the loop, it is highly adaptable to high-frequency trading, which is widely used because of the rapid increase of market volume and scale. Our project focuses on the simulator which simulates the financial market in reinforcement learning and puts out a novel agent-based simulator, Agent-Based Interactive Discrete Event Simulator (ABIDES)[1] which allows agents to interact with each other with their own messages and information of rest of the world.

Our primary achievement is the integration of Long-Short Term Memory (LSTM)[2] and ABIDES[3]. While traditional reinforcement learning follows the Markov decision process, which assumes that the current state is only related to the previous state, financial markets are influenced by multiple factors, and asset prices exhibit periodic trends. LSTM networks allow our model to remember the most relevant information from the past and present while discarding irrelevant data. Additionally, we use neural networks to define the state space[4], removing human intervention and including only the most important market elements to improve accuracy.

Our future research will focus on refining our model. The current action space is limited to simple actions such as buy, sell, and hold. We plan to make these actions more complex to better simulate financial agents. Furthermore, we aim to explain how our neural networks select states to uncover the underlying logic of these selections.

## 1.1 Introduction of the financial market with increasing trading volume and frequency

Ever since 2000, we have witnessed a soaring increase in trading volume. According to data from New York Stock Exchange (NYSE), trading volume grew by over 350% between 2002 and 2008[5]. Compared to the market before the 21st century, the financial market with larger volumes became more complicated

and varied for agents to trade and analyze. As a result, trading strategies must evolve in order to be efficiency and accuracy in the more complicated environment of today' s market. At the same time, with the improvement of calculating speed and dimension, computer-based trading methods were introduced where computer science was integrated with financial trading. Among them, high frequency trading, known as HFT, which is characterized by the high speed of orders, high turnover rates, and high order-to-trade ratio came to our attention. Around 2004, HFT had an execution time of several seconds, whereas by 2010 this had decreased to milli- and even microseconds, which makes the trade even faster[6].

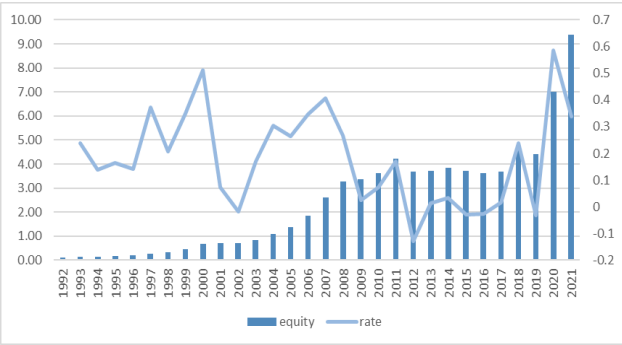


Fig 1. Trading Volume and Growth Rate

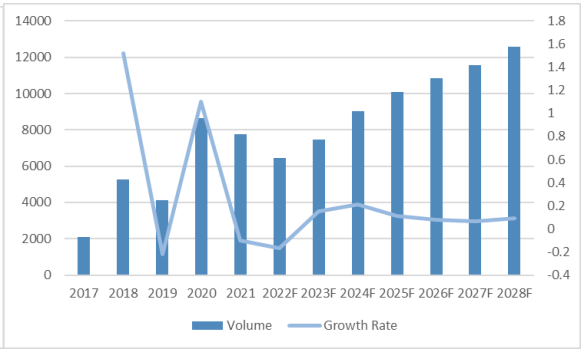


Fig 2. HFT Volume and Growth Rate Prediction

Traditional analyzing methods cannot effectively leverage large amounts of data in a highly complex market. Some suggested using machine learning which developing algorithms that can learn from data and make predictions or decisions without being explicitly programmed. One of the challenges in traditional machine learning is that training these algorithms require large quantities of labeled data. Additionally, traditional machine learning algorithms are limited in their ability to learn from experience and adapt to changing situations. Reinforcement learning is a subfield of machine learning that addresses these limitations by allowing algorithms to learn from experience and adapt to changing situations. In reinforcement learning, an agent interacts with an environment and receives feedback in the form of rewards or penalties based on their actions. The goal of the agent is to learn a policy that maximizes its cumulative reward over time[7].

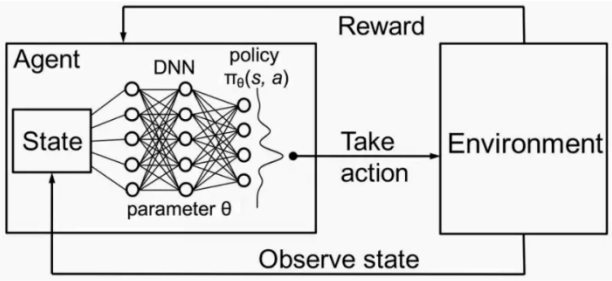


Fig 3. Mechanism of Reinforcement Learning

However, it is expensive to model the states and actions transition steps. Our project focuses on how to successfully simulate the whole financial market which not only guarantees accuracy but also is easily modelled. We use the OpenAI Gym framework to simulate a direct environment for reinforcement learning. What distinguishes our work from others is that we use ABIDES, Agent-Based Interactive Discrete Event Simulator, to separate each agent, consider his own Markov Decision Process (MDP), and regard the other agents as the rest of the world's background. Each agent interacts with others via the messages and takes actions that will be sent to the environment and returned a reward and new state. In this process, ABIDES is combined with OpenAI Gym [8], in which agents interact individually through a message system. Each agent only has his own state and information about the rest of the world. On the other hand, using machine learning to determine the states utilized in deep Q learning is also an effective way to solve the problem.

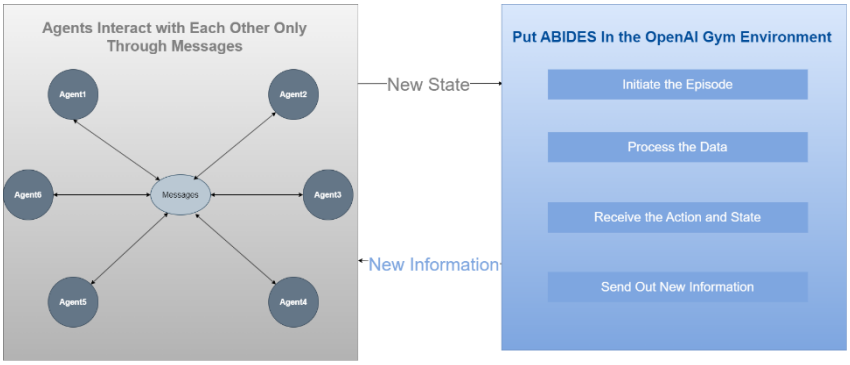


Fig 4. Mechanism of ABIDES

## 1.2 Capstone Objective: increase trading policy efficiency and profitability

Our project aims to combine reinforcement learning and the financial market by using ABIDES-GYM to simulate the market and agents. We have introduced new methods to enhance the training of our simulator, with the ultimate goal of improving the efficiency and profitability of trading strategies for both the market and financial agents.

Our result showed a significant improvement in the profit and episode reward of the trading policy using ABIDES and Proximal Policy Optimization (PPO) compared to other trading policies. We observed a 2.38% profit margin after 700 iterations of training, which exceeds other policies which have negative profits. We also witnessed that with a higher learning rate, the trading policy we proposed performed better which is unnormal in other training processes. The explanation is we used PPO which uses the new collected data and focused on the latest policy, as known as on policy, compared to other methods, off policy. The lower learning rate may cause overfitting in our training and perform worse.

Table 1. Profit of Different Policies

| name | episode_reward | Profit | Profit_Margin | cash | holdings | spread | marked_to_market |
|---|---|---|---|---|---|---|---|
| ppo: lr=0.01,seed=1 | 6.114810 | 23847.76 | 2.3848% | 9.830686e+05 | 0.4 | 1.58 | 1023847.76 |
| passive | 0.000000 | 0.00 | 0.0000% | 1.000000e+06 | 0.0 | 2.04 | 1000000.00 |
| dqn | -0.051795 | -202.00 | -0.0202% | 4.204756e+05 | 5.8 | 1.70 | 999798.00 |
| random_no_action | -2.100179 | -8190.70 | -0.8191% | 1.601430e+06 | -6.0 | 1.40 | 991809.30 |
| random | -6.835390 | -26658.02 | -2.6658% | -1.894300e+08 | 1904.2 | 1.72 | 973341.98 |
| aggressive | -16.317056 | -63636.52 | -6.3637% | -3.729324e+08 | 3738.8 | 2.18 | 936363.48 |

Table 2. Performance Using Different Learning Rates

| name | episode_reward | Profit | Profit_Margin | cash | holdings | spread | marked_to_market |
|---|---|---|---|---|---|---|---|
| ppo: lr=0.01,seed=1 | 6.114810 | 23847.76 | 2.3848% | 9.830686e+05 | 0.40 | 1.58 | 1023847.76 |
| ppo: lr=0.0001,seed=3 | 3.504077 | 13665.90 | 1.3666% | -3.445695e+06 | 44.60 | 1.40 | 1013665.90 |
| ppo: lr=0.0001,seed=1 | 3.361390 | 13109.42 | 1.3109% | -1.686267e+06 | 27.00 | 1.54 | 1013109.42 |
| ppo: lr=0.001,seed=1 | 2.035990 | 7940.36 | 0.7940% | -3.195727e+06 | 42.00 | 1.82 | 1007940.36 |
| ppo: lr=0.01,seed=3 | 2.002508 | 7809.78 | 0.7810% | -1.900177e+08 | 1910.00 | 1.56 | 1007809.78 |
| ppo: lr=0.0001,seed=2 | 1.263800 | 4928.82 | 0.4929% | -2.273719e+07 | 237.50 | 1.66 | 1004928.82 |
| ppo: lr=0.001,seed=3 | 0.367159 | 1431.92 | 0.1432% | 6.513820e+06 | -55.20 | 1.50 | 1001431.92 |
| ppo: lr=0.001,seed=2 | -5.515923 | -21512.10 | -2.1512% | 1.074026e+08 | -1063.80 | 1.82 | 978487.90 |
| ppo: lr=0.01,seed=2 | -7.896462 | -30796.20 | -3.0796% | -3.577650e+08 | 3587.22 | 1.40 | 969203.80 |

# 2.Related work about methods used in the capstone project

## 2.1 Reinforcement Learning to illustrate how agents choose strategies

Reinforcement learning[9] is a method to illustrate how smart agents choose their strategy to optimize their objectives. In RL, the environment is typically stated in the form of a MDP[10]. There are two types of RL, model-based and model-free. Model-based methods assume the information needed to construct MDP, while model-free methods assume the information is unknown. To gain more information about states, actions, and rewards, agents need to interact with the environment. Since simulations are leveraged to avoid the high cost of training a model through direct interactions with real-world system.

## 2.2 ABIDES multi-agent simulation environment to create artificial stock market

Among plenty of simulators, ABIDES is utilized because it provides agents with their own message and information about the rest of the world. Agents are divided into the exchange agent, the agent in simulation, and market participants, other agents[1]. They communicate only by messages, which include orders and market data. However, it requires a deep understanding of the agents and enviroments. Besides, the framework is unconventional to use the existing model.

## 2.3 OpenAI Gym to develop and test learning agents

OpenAI Gym is an environment for developing and testing learning agents. It is a focused and best-suited reinforcement learning agent [11]. To avoid the drawbacks of ABIDES, we use it through the OpenAI Gym environment framework. Running ABIDES as a black box and leaving the learning and MDP formulation process outside the simulator, we overcome the obstacles we met when using ABIDES alone[1]. Besides, we combine ABIDES-Gym and ABIDES-Markets to build ABIDES-Gym-Markets, an abstract base environment that has the functionality of both platforms. In addition, by leveraging open-sourceRL tools, we demonstrated that an RL agent could easily be trained using ABIDES-Gym-Markets[12].

# 3.Baseline Model: ABIDES-Gym Environment and Deep Q-Learning

## 3.1 Introduction about Deep Q Learning: feeding data in a continuous simulated market

To train RL agents to trade in the simulated market environment, we propose using the Deep Q-Learning[13] (DQN) algorithm for the baseline model, as DQN is easy to implement and can easily adapt to a simulated realistic market environment, such as the ABIDES-Gym. Large-scale and complex datasets can be directly fed into DQN for training.

More specifically, we use the simulated data as input to feed into the neural network architecture in the DQN model, where the neural networks will compute the rewards or profits the trading agents obtained for each trading action they make at different market states. To train the neural network toward our objective, we calculate the loss function by comparing the values to the Bellman equation and updating the weights of the neural network via stochastic gradient descent (SGD) and backpropagation (BP). The more training iterations we perform, the better the DQN model is good at outputting optimal values, which leads to an optimal policy in the action space.

To build a DQN model, we use a fully connected feed-forward neural network with 2 layers composed of 50 and 20 neurons. We implement an epsilon-greedy search approach decreasing from 1 to 0.02 in 10k steps to allow the agents to interact with the environment fully. The learning rate is scheduled to linearly decrease from $1 \cdot 10^{-3}$ to 0 in 90k steps.

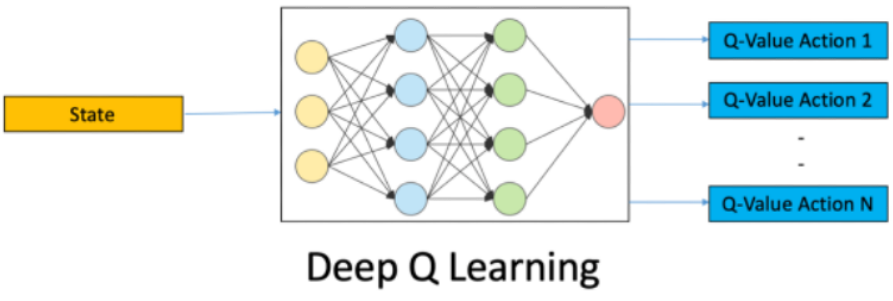


Fig 5. Representation of a Deep Q Learning model

After simulating the synthetic data and choosing a Deep RL model for the trading agents, our next challenge will be to simulate the simultaneous training process of multiple trading agents in the ABIDES environment. ABIDES is designed to support AI agent research in market applications, by enabling the simulation of tens of thousands of trading agents, which interact with an exchange agent to receive information about the market, to facilitate transactions.

## 3.2 ABIDES-GYM associated with Deep Q Learning

We aim to simulate our agents in the ABIDES-Gym environment which utilizes the OpenAI Gym framework, an open-source Python library used for developing and comparing reinforcement learning algorithms by providing a standard API to communicate between learning algorithms and environments. Unlike the original ABIDES, the MDP formulator is left outside of the simulator and can communicate with the agents. Thus, the user can interrupt the simulation by sending out a request, unlike the original environment in which the whole simulation plays out without interruption. When paused, the system returns the current state of a specified agent. After being processed using Kernel methods, this data can be used to run further computations or be fed into the DQN algorithm to define the next action. To use the ABIDES-Gym environment, the user can then use the runner method with action a as input to resume the simulation until the next interruption.

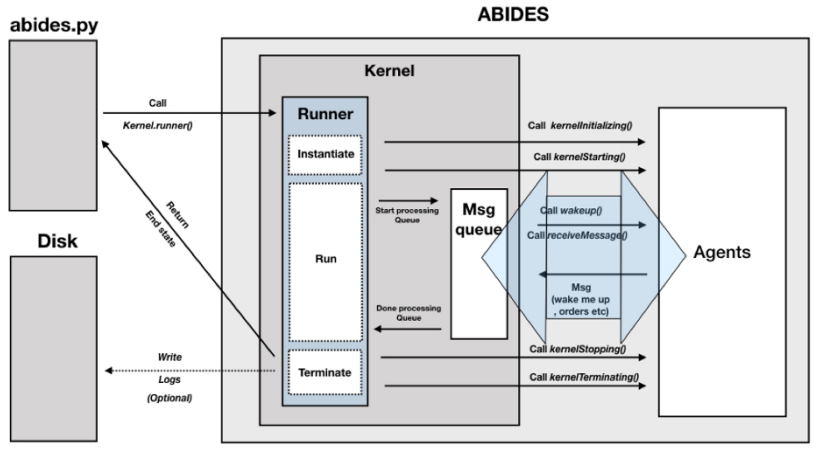


Fig 6. ABIDES-Gym Environment

This environment presents us with a way to accurately simulate our agents and have great control over the data that we use as input for our model. Using it is a great opportunity and can play a major role in the overall success of our project.

## 3.3Markov Decision Process: States, Actions and Rewards

Our reinforcement learning agent operates by taking decisions in a continuous environment following a Markov Decision Process (MDP).

A Markov Decision Process (MDP) is a mathematical framework utilized in various fields, including finance and economics, to model decision-making problems. It serves as the foundation for reinforcement learning, a type of machine learning algorithm that allows an agent to learn from its experiences and make improved decisions over time[14]. In a continuous financial market, an MDP can be used to train a reinforcement learning agent to optimize trading strategies.

A Markov Decision Process is defined by a state space, an action space and a reward function which is associated with the decision taken in a given spate (t) as a function of only the last recorded state (t-1).

### *3.3.1 State Space:*

The state space comprises all possible states an agent can be in at any given time. In a continuous financial market, this space includes both private states, known only to the agent, and public market states. At each time step, trading agents perceive the market through the following states:

- $remainingTime_t = \frac{T-t}{T}$: the time remaining at time $t$;
- $remainingQuantity_t = \frac{M-m}{M}$: the number of stock shares remaining at time $t$, where $M$ is the total amount of shares the agent must trade, and $m$ is the amount of shares the agent has traded up to time $t$;
- $spread_t = bestAsk_t - bestBid_t$: the difference between the the best bid price and the best ask price at time $t$;
- $imbalance_t = \frac{totalBidVolume_5}{totalBidVolume_5 + totalAskVolume_5}$: the difference between the order volume on the bid side and the ask side of the first 5 levels;
- $R_t^k = (r_1, \cdots, r_k)$: a series of mid-price returns, where $r_j = log(\frac{midPrice_t}{midPrice_{t-j}})$ and $k = 5$;

- $priceImpact_t = midPrice_t - midPrice_0$: the difference between the mid price at time $t$ and the mid price at time $T_0$;
- $marketOrderCost_t$: the best bid price at time $t$ if the agent is selling or the best ask price at time$t$ if the agent is buying;
- $signedTransactionVolume_t$: the signed volume (buy orders are positive and sell orders are negative) within the past time period at time $t$;

### *3.3.2 Action Space:*

The action space consists of the possible actions an agent can take at each time step. In our continuous financial market context, the action space includes the following actions:

- 'Buy': The agent buys a certain quantity of shares.
- 'Sell': The agent sells a certain quantity of shares.
- 'Hold': The agent neither buys nor sells any shares.

### *3.3.3 Reward Function:*

The reward function measures the success of the agent's actions and guides its learning. In this setting, the reward function is defined as:

$$R(t) = \sum_{o \in O_t} \epsilon(a).(transactionPrice_{0,t} - midPrice_0).quantity_{o,t}/M$$

Here, $O, t$ represents the set of executed orders at time t, $\epsilon(a)$ is 1 for buy orders and else is -1, $transactionPrice_{0,t}$ is the executed price for order o at time t, $midPrice_0$ is the mid-price at the initial time step, and $quantity_{o,t}$ is the executed volume for order o at time t.

The reward function captures the impact of executed orders on the agent's performance relative to the initial mid-price, normalized by the total number of shares the agent must trade (M).

Furthermore, we add a Reward associated to the episode to place a penalty based on the amount of uncompleted trades:

$$R(episode) = max(penalty\,.(M - m_{T_N}), 0)$$

Where $penalty = 100$ per shares

In summary, our reinforcement learning agent follows a markov decision process and each reward serves as a feedback to update the model and make it better.

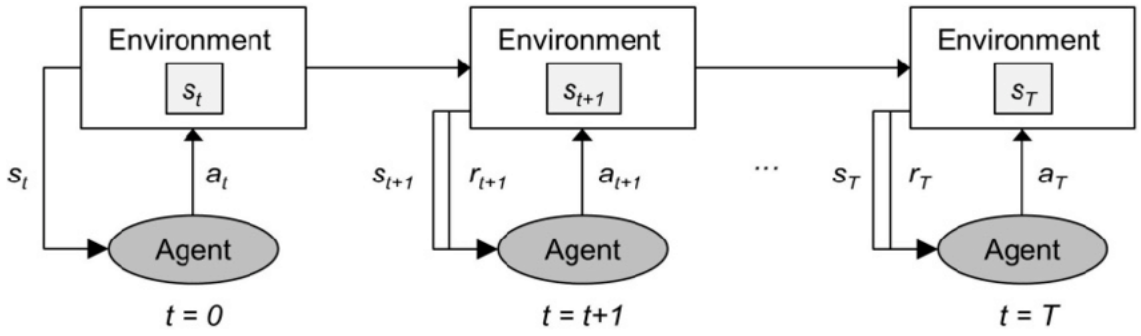


Fig 7. Reinforcement Learning Design

The agent operates in a continuous environment and his goal is to find the most optimal action to take in each state, action that will maximize its reward at time t.

# 4.Proposed Modeling Techniques with LSTM and PPO

In our proposed model, we first propose to add LSTM (Long Short-Term Memory) layers to train trading agents in the DQN algorithm. One limitation of the current DQN model is that it is only able to utilize the current input information and hence when it is applied to time series data, the algorithm fails to make use of agents' experience learned from previous data.

## 4.1 LSTM Implementation in Deep Q Learning to process time series data

On the other hand, LSTM is known for its remarkable performance in processing time series data. By incorporating historic information from states far in the past, trading agents can take advantage of

previous experience, and therefore the predicted Q-values and the optimized trading policies will be better.

Built on classical recurrent neural networks (RNN) structure, LSTM mildly modifies the structure to incorporate three gates, input gate, output gate and forget gate, to handle the vanishing gradient problem, where RNN begins to forget information far in the past. With these gates, LSTM is able to distinguish between important and less important information. The structure of LSTM can be shown in Figure 5.

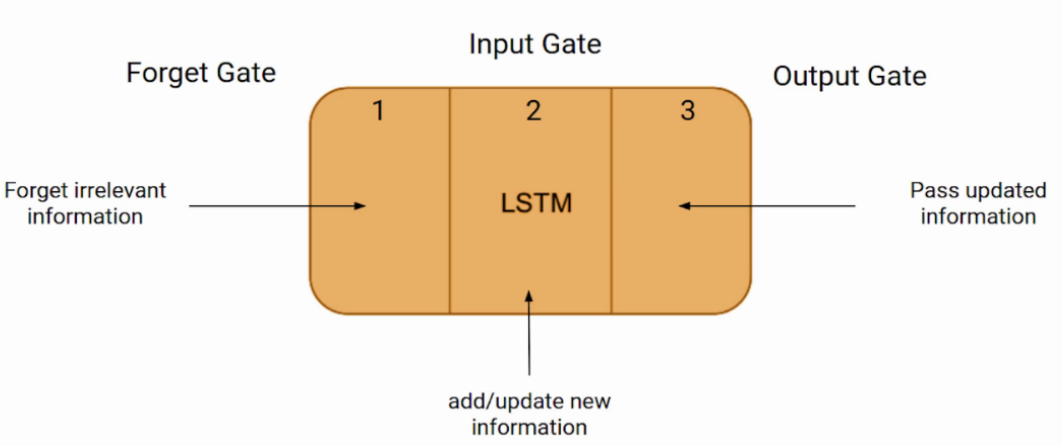


Fig 8. LSTM Architecture

## 4.2 PPO implementation to update policy based on old and new policies information

Apart from LSTM, we also think about incorporating Proximal Policy Optimization (PPO). PPO is a deep reinforcement learning algorithm commonly used to train agents. It is an on-policy algorithm, meaning that it updates policy based on the data collected in the interactive environment under the current policy. PPO works by optimizing a surrogate objective function that approximates the expected improvement in the policy. The surrogate objective function is designed to be easy to optimize while still being a good approximation of the true objective. The core idea of PPO is to limit the size of the policy updates at each iteration, which can make sure that the policy does not change too much and cause instability. This is done by a clipping mechanism to ensure that the policy update does not deviate too far

from the previous policy. Besides, PPO also uses the observations to update from the old policy to the new policy. The architecture of PPO is shown in Figure 6.

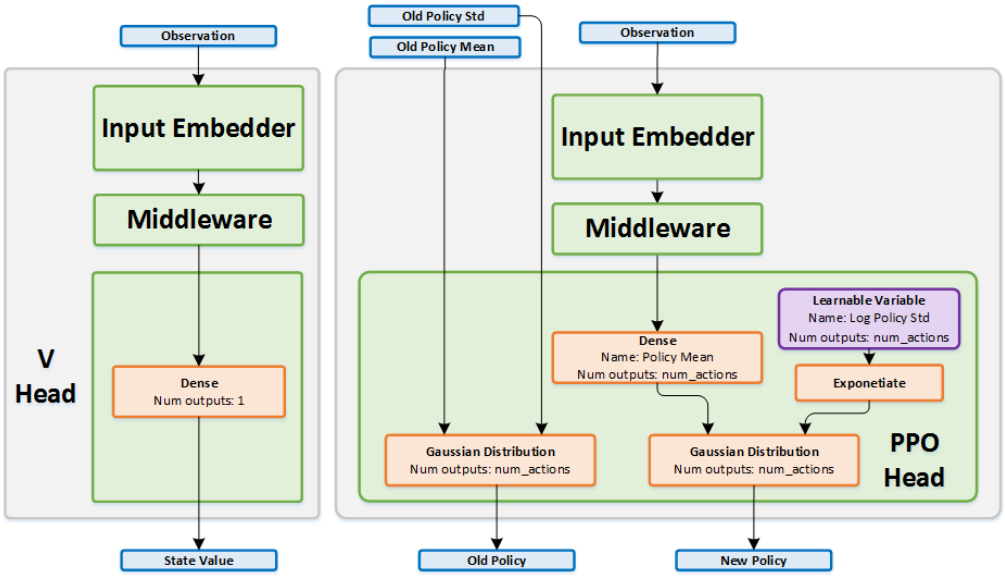


Fig 9. PPO Architecture

PPO has been shown to perform well on a wide range of tasks, including the daily investor problem. One of the advantages of PPO over DQN is that PPO is an on-policy algorithm while DQN is an off-policy algorithm, making PPO more sample-efficient and less prone to overfitting, especially in a complicated and noisy environment such as the financial market. Besides, PPO uses a surrogate objective function, making it more stable and robust to changes in the environment, while DQN uses the Bellman equation to update Q-values, which can lead to instability in the policy and divergence in some cases.

Experiment results have shown that PPO generally performs better than DQN and can achieve convergence in a relatively fewer number of steps. PPO is also more suitable to implement in the ABIDES-Market environment, which requires exploration and exploitation to fully capture the market complexity. On the other hand, DQN relies on the epsilon-greedy exploration strategy, which may not be

effective in a complex and high-dimensional environment like the financial market. To better compare the performance of PPO with DQN, we use the same neural network architecture as DQN. [Jingqi, Charlotte]

## 5.Simulated Stock Market Analysis

In this section, we present the analytical results of the simulated stock market as well as the behavior of the trading agents. The simulated daily investor environment includes 1 exchange agent, 2 adaptive market maker agents, 1,000 noise agents, 102 value agents, and 12 momentum agents, where the exchange agent is responsible for the update of the limit order book at different timesteps, the adaptive market maker agent helps ensure the liquidity of the simulated market, the noise agent randomly buys or sells the stock shares, the value agent buys or sells the stock based on the most recent stock price, and the momentum agent makes trading decisions based on the stock price movement in a given period.

### 5.1 Stock Price and Order Volume: close to real-world financial market

In this section, we analyze the simulated stock price and trading volume to validate whether the stock market closely resembles the real-world financial market.

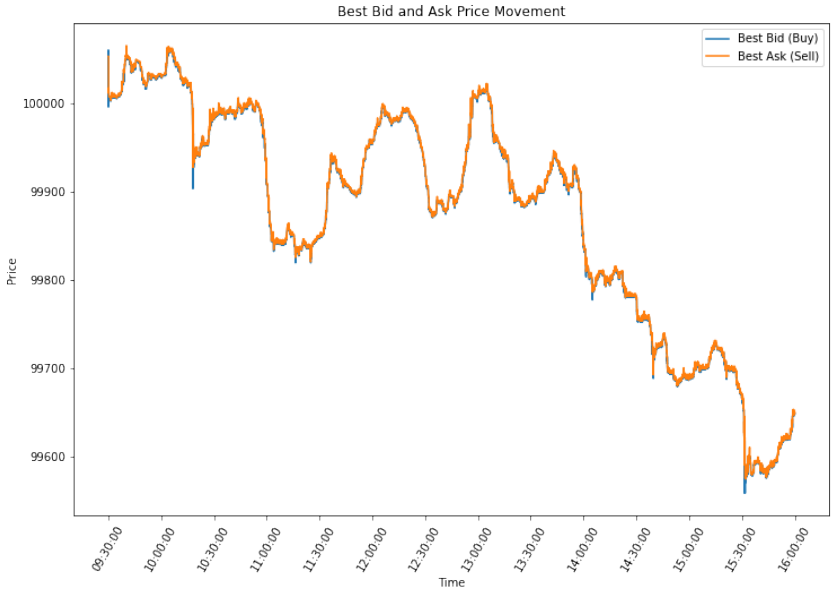


Figure 10. Best Bid Price and Best Ask Price Movement

Figure 7 shows the best bid and best ask price movements from 9:30 a.m. to 4:00 p.m. It shows that the price movement closely resembles the real-world stock price movement with several fluctuations and an overall downward movement. The overall downward trend represents the one-day behavior of a stock while the sharp changes in the stock price represent macro shocks to this stock.

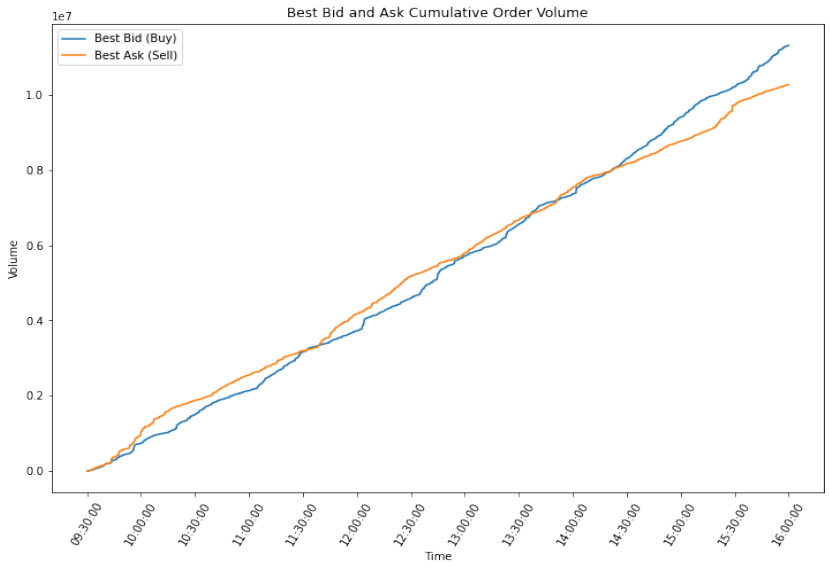


Figure 11. Cumulative Order Volume of Best Bid and Ask

Figure 8 presents the cumulative order volume or order quantity for the whole day. It shows that the order volume is approximately linearly increasing with respect to time and at the end of the day, it will reach about 12 million, indicating that there will be more than 10 million orders being executed in a single day for just one stock.

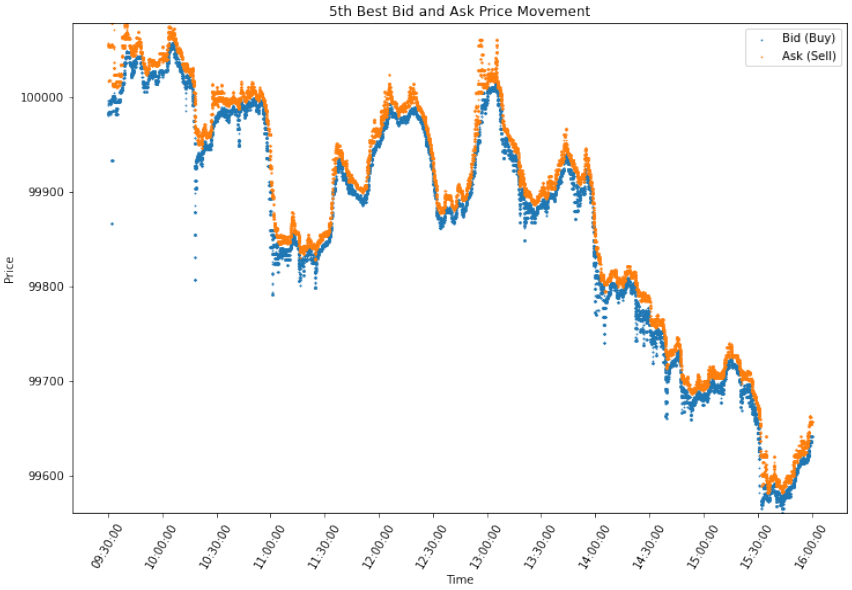


Figure 12. 5th Best Bid and Best Ask Price Movement

We also include the bid and ask price movement from the 5th level to utilize more information on the stock market. Generally, the stock price movement is the same as the best bid and best ask, while the bid and ask price would have more small fluctuations along the day.

## 5.2 Trading Agent Behavior for different trading agents

We also analyze the behavior of the trading agents, including their order submission time distribution and trading order quantity and price distribution to ensure that agent behavior is reasonable.

Figure 10 shows the number of submitted orders across time for value agents, where the number of orders submitted is similar at different times of the day. Figure 11 shows the number of submitted orders across time for momentum agents, where the number of orders submitted increase by 100 in the first hour and stays relatively stable throughout the rest of the day, and there is also a slight decrease in the last 30 minutes.

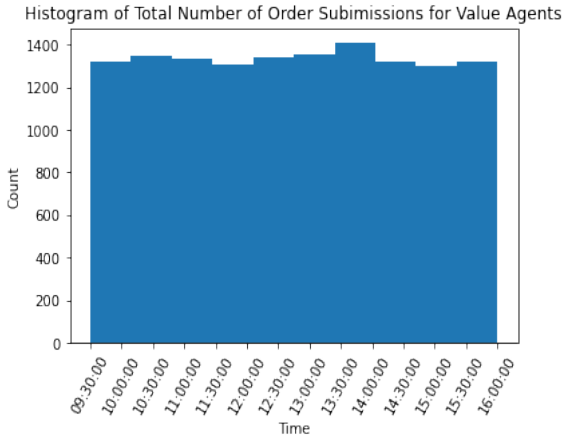


Figure 13. Histogram of Number of Order Submissions Across Time for Value Agents

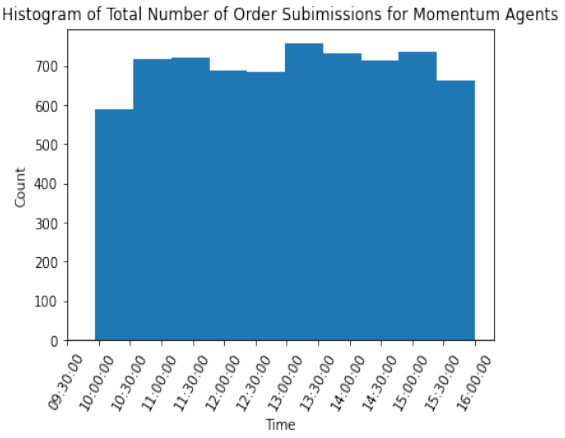


Figure 14. Histogram of Number of Order Submissions Across Time for Momentum Agents

The behavior of value agents and momentum agents in general agree with the design of these two types of agents, where the value agent makes trading decisions based only on the most recent stock price, which does not change much throughout the day, while the momentum agent makes trading decisions based on a series of recent stock prices. The behavior of the momentum agent also represents a smart trading strategy, i.e., to place fewer orders when the market is unstable.

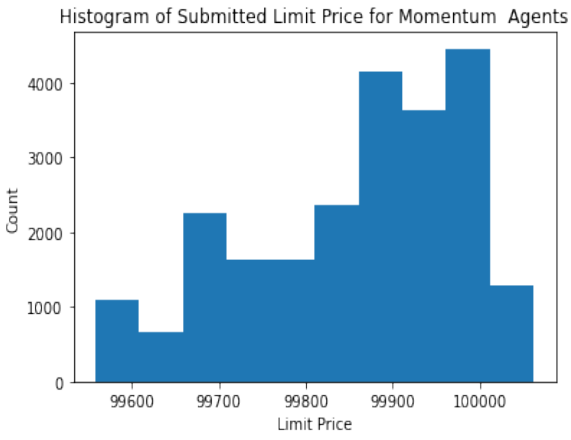


Figure 15. Histogram of Submitted Price for Value Agents

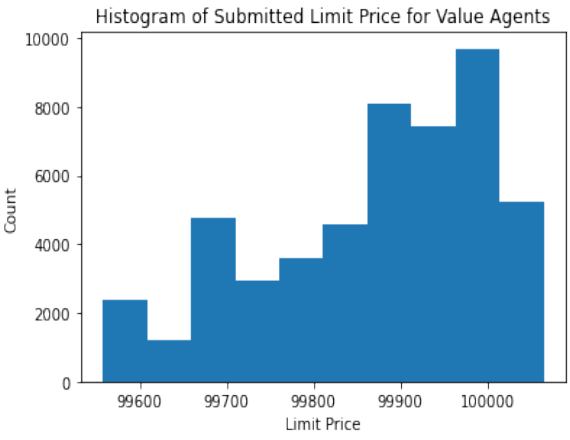


Figure 16. Histogram of Submitted Price for Momentum Agents

Figure 12 and Figure 13 show the histograms of the trading price for value agents and momentum agents. In general, the two types of agents trade at similar prices, except that the value agents tend to place more orders at the highest price.

# 6.Preliminary Results about Reinforcement Learning Model

## 6.1 General Research Steps for model training and testing

Our project wrote and implemented the python script for reinforcement learning model training by referencing the demo of JP Morgan ABIDES gym on the github. First, we constructed a simulated stock market, where investors try to interact with each other and make money buying and selling a stock,with in-depth trading data using the daily investor environment in ABIDES.

Besides, we also introduced the Ray Library into our project, which enables easy parallelization for our reinforcement learning model by doing a grid search on the hyper-parameters, such as the learning rate. Using Ray, we can easily run the deep Q learning model, LSTM, as well as the PPO model with our previously simulated stock market. In order to better understand the training progress and conduct hyper-parameter search, we use the Ray Dashboard and Tensor Board to track the progress. After training for about 1000 episodes and reaching convergence, we can get an optimal trading policy in the context of the current model configuration about states, actions, and rewards.

With the optimal trading strategy, we further simulated 50 episodes to trade using this strategy proposed by the reinforcement learning algorithm. To relatively evaluate its performance, we further introduced some simple trading strategies such as a passive policy where there is no transactions at all, an aggressive policy where always buying stocks, a random policy where the actions of whether buy or sell are chosen randomly, as well as a random policy including hold where the actions of whether buy, sell or hold are chosen randomly. We compared the overall results of all five trading policies.

## 6.2 Current results under training of 100 episodes

Currently, we only run the python script on our local computers. Due to the limited computational power, we can only train the reinforcement learning model by 100 episodes under one hyper-parameter setting. Under 100 episodes, we can get the following result:

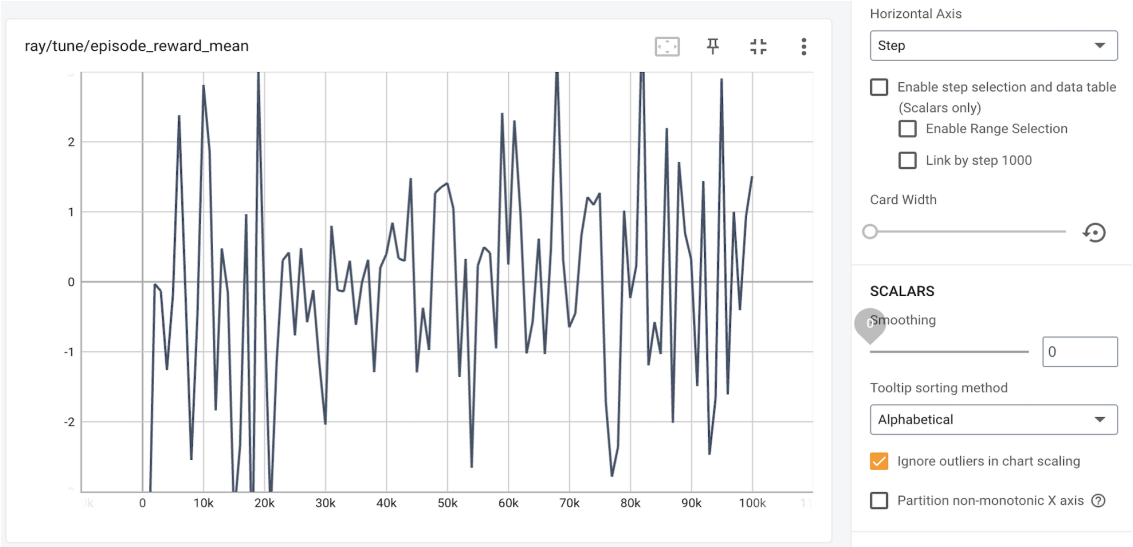


Figure 17. Training average accumulated rewards trend

Looking at the average episode rewards, they are still quite random after 100 training and don't have any signs of reaching convergence. This means that the current training episodes are not enough. In order to solve this problem, we are trying to use the SAVIO cloud computational units to implement more training.

Table 3. Training results comparison for five different policies

| name | episode_reward | cash | holdings | spread | marked_to_market | profit | profit_percentage |
|---|---|---|---|---|---|---|---|
| **passive** | 0.00 | 1.000000e+06 | 0.00 | 2.04 | 1000000.00 | 0.00 | 0.00 |
| **random_no_action** | -1.31 | -7.334779e+06 | 83.26 | 1.40 | 969258.68 | -30741.32 | -3.07 |
| **aggressive** | -8.66 | -1.144636e+09 | 11444.26 | 2.94 | 797352.94 | -202647.06 | -20.26 |
| **random** | -11.23 | -7.926926e+08 | 7931.80 | 5.74 | 737124.24 | -262875.76 | -26.29 |
| **r1** | -13.39 | -1.076737e+09 | 10768.38 | 2.72 | 686777.98 | -313222.02 | -31.32 |

For the comparison among different policies, the preliminary results are shown in the following table. We can find that currently the performance of the reinforcement learning model is not very ideal.

Further analyzing the distribution of all (cumulated) episode rewards and profit_percentage for all 50 training, the results are shown in the following graphs.

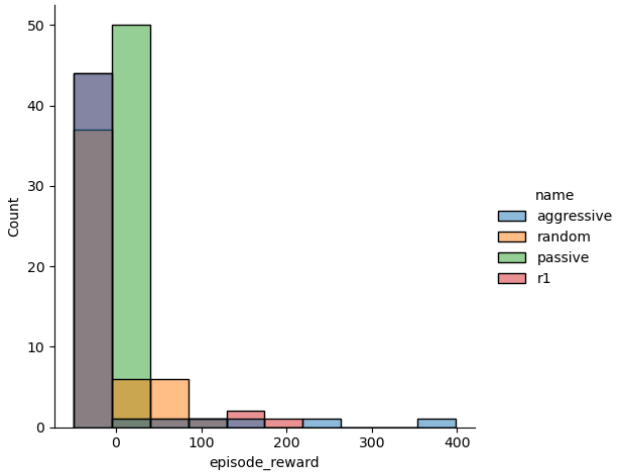


Figure 18. Histogram of average accumulated rewards

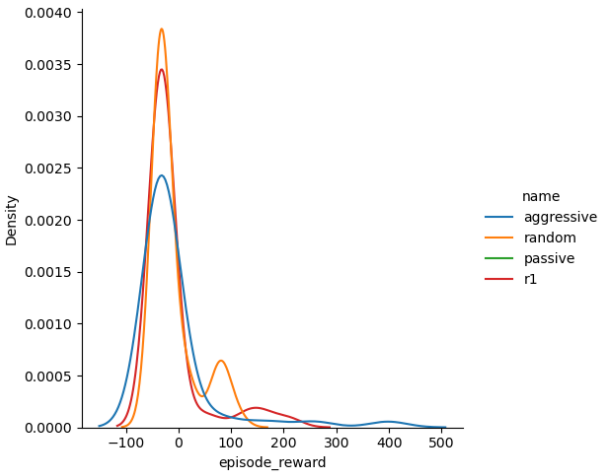


Figure 19. Distribution of average accumulated rewards

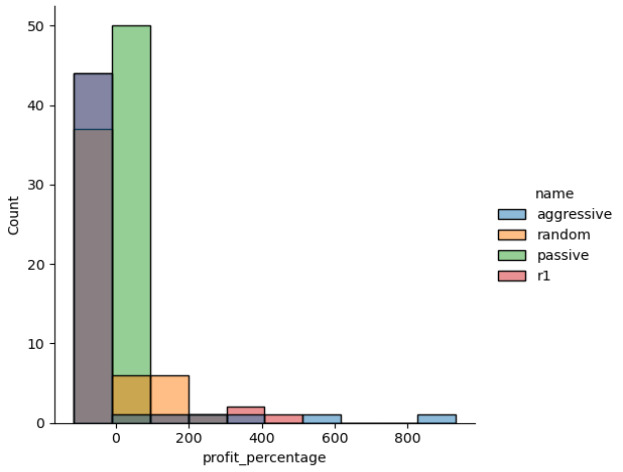


Figure 20. Histogram of average profit percentage

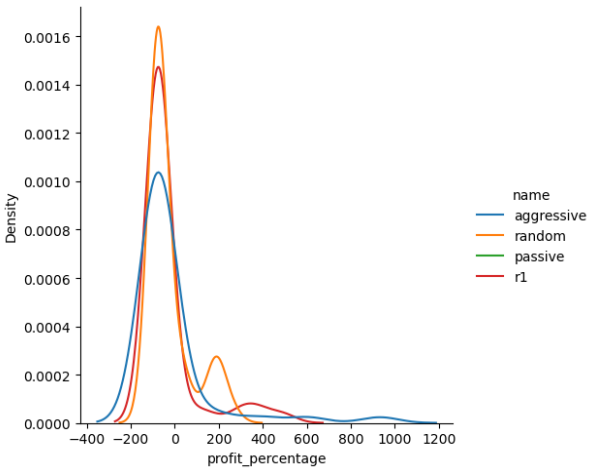


Figure 21. Distribution of average profit percentage

Currently the distribution of reinforcement learning is quite similar with other three policies. (Here we didn't include the 'random_no_action' policy because its results are highly distributed around 0, which makes the comparison among the other four policies not visible.)

# 7.Conclusion and future steps to improve profitability

Our project utilizes a new method in reinforcement learning named ABIDES which is a simulator where agents interact with each other with messages and take actions according to the state and information they have. We train it in the OpenAI gym to simulate the whole financial market. Revolutionarily, we combine LSTM with ABIDES to let the simulator memorize more previous information which simulates the financial market concisely. We also use neural networks to determine the state space in order to keep humans out of the loop further. In the future, we are going to keep on going to enlarge the action space as well as doing explanatory research in the state space selected by the neural network.

# References:


[1] Selim Amrouni, Aymeric Moulin, Jared Vann, Svitlana Vyetrenko, Tucker Balch, Manuela Veloso, ABIDES-Gym: Gym Environments for Multi-Agent Discrete Event Simulation and Application to Financial Markets, https://arxiv.org/abs/2110.14771

[2] Karima Amoura, Patrice Wira, Said Djennoune, A State-space Neural Network for Modeling Dynamical Nonlinear Systems, https://www.scitepress.org/papers/2011/36805/36805.pdf

[3] David Byrd, Maria Hybinette, Tucker Hybinette Balch, ABIDES: Towards High-Fidelity Market Simulation for AI Research, https://arxiv.org/abs/1904.12066

[4] JózefKorbicz, MarcinMrugalski, ThomasParisini, Designing State-space Models With Neural Networks, https://doi.org/10.3182/20020721-6-ES-1901.01630

[5] Trading Volume from 1992 to 2021, According to NYSE, https://www.nyse.com/trading-data

[6] High Frequency Trading Market Report, https://dataintelo.com/report/high-frequency-trading-market/

[7]Olivier Buffet, Olivier Pietquin, Paul Weng, Reinforcement Learning. https://arxiv.org/abs/2005.14419

[8] Svitlana Vyetrenko, David Byrd, Nick Petosa, Mahmoud Mahfouz, Danial Dervovic, Manuela Veloso, Tucker Hybinette Balch, Get Real: Realism Metrics for Robust Limit Order Book Market Simulations, https://arxiv.org/abs/1912.04941

[9] Zihao Zhang, Stefan Zohren, Stephen Roberts, Deep Reinforcement Learning for Trading, https://arxiv.org/abs/1911.10107

[10] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, Martin Riedmiller, Playing Atari with Deep Reinforcement Learning, https://arxiv.org/abs/1312.5602

[11] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, Wojciech Zaremba, OpenAI Gym, https://arxiv.org/abs/1606.01540

[12] Michaël Karpe, Jin Fang, Zhongyao Ma, Chen Wang, Multi-Agent Reinforcement Learning in a Realistic Limit Order Book Market Simulation, https://arxiv.org/abs/2006.05574

[13] Hui Niu, Siyuan Li, Jian Li, MetaTrader: An Reinforcement Learning Approach Integrating Diverse Policies for Portfolio Optimization, https://arxiv.org/abs/2210.01774

[14] Tohid Atashbar, and Rui (Aruhan) Shi, Deep Reinforcement Learning: Emerging Trends in Macroeconomics and Future Prospects, https://www.elibrary.imf.org/downloadpdf/journals/001/2022/

259/001.2022.issue-259-en.pdf