Computer Science
&
Data Science

Capstone Report - Spring 2022

Momentum Strategy with Deep Reinforcement Learning in Chinese Stock Market

Eric Yang,
Oliver Wang,
Jerry Shi

supervised by
Christina Wang

Preface

Zhangyi Wang majors in Economics and Data Science. He has worked extensively in econometrics and machine learning methods, and displays strong interest in quantitative finance. Zhihan Yang majors in Data Science and Business and Finance. He has both Data Science knowledge and skills in financial analysis. Ruijie Shi majors in Data Science with a finance concentration. He developed an interest in data analytics. With a background in both Finance and Data Science, we are eager to explore the application of algorithmic approaches in trading. We were inspired by the idea of "momentum", which seems to be intuitive while turns out to sophisticated. How can we define a momentum and how we can capture such momentum within a certain market and among all financial signals remains a riddle for both financial analysts and quantitative researchers. With the support of Professor Christina Wang, we obtained access to the first open-source DRL framework, FinRL. Based on this framework, we attempt to implement momentum strategies using Deep Reinforcement Learning. It is the first trial to combine a momentum strategy with deep reinforcement learning and we hope the results and conclusions of this paper can provide some valuable insights for traders who are interested in algorithmic trading. The ultimate goal of this paper is to show what beginners in quantitative finance can do with the complicated algorithms to design their own strategies to trade, inspiring more people to join us and explore the market with their own perception and intelligence.

Acknowledgements

We gratefully acknowledge the assistance of Professor Christina Wang, who guides us throughout the project and offers valuable insights towards quantitative finance. We are also grateful to Professor Li Guo, who provides us with detailed instructions and comments about every assignments. In addition, We would like to extend our gratitude to NYU Shanghai library faculty who assist us in getting access to the Wind financial terminal and helped us with the acquirement of the data.

Abstract

Momentum strategy is among the most popular strategies applied by traders and investors. The intuition behind the strategy is to buy "winners" and sell "losers". With the development of machine learning approaches, plenty of research has been done to examine the profitability of momentum strategy in learning-based automatic trading. Based on the first open-source DRL framework, FinRL. We study a risk-adjusted momentum strategy using the DDPG (Deep Deterministic Policy Gradient) model. Our research distinguishes from previous studies on learning-based momentum strategy in the following aspects: first, we apply DRL algorithms based on the first open-source DRL framework, FinRL to do automatic trading. It will be the first attempt to use deep reinforcement learning to optimize momentum strategy. Second, we focus on the Chinese stock market instead of the commonly used future market in the U.S. We work on SSE 50 Index constituents stocks which serve as an important reference for the performance of momentum strategies within the Chinese market. Back-testing of our strategy shows that momentum strategy performs well in the Chinese stock market, beating the buy&hold as well as the daily optimization benchmarks. A potential source of such excellence might be the longer persistence of trend within the market.

Keywords

Momentum Strategy; DDPG; FinRl; Stock Market

Contents

1. Introduction

Momentum strategies are derived from the philosophy that strong price trends have a tendency to persist[1]. The most important assumption of momentum strategies lies in that winners will continue to be winners in the subsequent period. Plenty of research has been done to quantify the magnitude of such trends and to size traded positions accordingly. A key problem that needs to be addressed concerning momentum trading is trend reversal[1]. Can a strategy be created that avoids the process of momentum toward reversal and obtains higher risk-adjusted returns? We attempt to answer this question using advanced machine learning techniques as well as quantitative financial theories and models.

AI community has accumulated an open-source code ocean over the past decade. Numerous attempts have been made to apply machine learning approaches to design efficient trading strategies. Essentially, the process of trading can be described as an dynamic decision-making problem involving two critical steps: market conditions capturing and optimal action execution[2]. Compared with conventional machine learning tasks, dynamic decision-making is more challenging due to the lack of supervised information. It requires the researcher to explore an unknown environment all by itself and to simultaneously make correct predictions and decisions in an online manner. A major challenge involved in applying algorithmic approaches to do automatic trading originates from the difficulties in financial environment representation. The financial data contain a large amount of noise, jump, and movement, leading to the highly non-stationary time series[2]. Such characteristics of the problem inspired researchers to shift from conventional machine learning methods to advanced deep learning and reinforcement learning algorithms which can deal with a complex environment where financial signals are hard to capture. Given such context, Deep reinforcement learning (DRL) has been recognized as an effective approach in quantitative finance[3]. With the development of modern architectures such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), deep learning models have been acknowledged for their ability to build representations of datasets, capturing temporal dynamics and cross-sectional relationships in a purely data-driven manner[4]. However, training a profitable DRL trading agent that decides where to trade, at what price, and what quantity involves error-prone and arduous code development and debugging. Therefore, utilizing existing frameworks and packages becomes an efficient method for implementing research and organizing the relevant codes and files in a systematic way.

Based on the first open-source DRL framework, FinRL. We study a risk-adjusted momentum strategy using the DDPG (Deep Deterministic Policy Gradient) model. Our research distinguishes from previous studies on learning-based momentum strategy in the following aspects: first, we apply DRL algorithms based on the first open-source DRL framework, FinRL to do automatic trading. It will be the first attempt to use deep reinforcement learning to optimize momentum strategy. Second, we focus on the Chinese stock market instead of the commonly used future market in the U.S. We work on SSE 50 Index constituents stocks which serve as an important reference for the performance of momentum strategies within the Chinese market. Back-testing of our strategy shows that momentum strategy performs well in the Chinese stock market, beating the buy&hold as well as the daily optimization benchmarks. A potential source of such excellence might be the longer persistence of trend within the market.

2. Related Work

The foundation of momentum as well as other other algorithmic trading strategies lies in the belief that stock prices do not follow random walks[5]. Jegadeesh et al.provided evidence of stock return predictability[5]. They found that negative first-order serial correlation in monthly stock returns is highly significant. Furthermore, a strong positive serial correlation was found at longer lags, and the twelve-month serial correlation was particularly strong[5]. To investigate the economic significance of the observed empirical regularity, ten portfolios were formed based on returns predicted using ex-ante estimates of the regression parameters. The results appeared quite striking and suggest that the extent to which security returns can be predicted based on past returns is economically significant[5]. The authors attributed the predictability of stock returns to market inefficiency and systematic changes in expected stock returns. Based on this assumption of market predictability, plenty of research has been done to capture the idea of "momentum" and design trading strategies that could be potentially profitable. The trading strategy that buys past winners and sells past losers was first examined by Jegadeesh et al.[6], which realized significant abnormal returns over the 1965 to 1989 period. The researchers provided an analysis of relative strength trading strategies over 3 to 12-month horizons. The results indicated that the profits are not due to the systematic risk of the trading strategies [6]. In addition, the evidence indicated that the profits cannot be attributed to a lead-lag effect resulting from delayed stock price reactions to information about a common factor. Instead, the evidence was consistent with

delayed price reactions to firm-specific information[6]. This paper serves as a foundation for the upcoming studies on momentum strategy and sheds light on the feasibility and profitability of such strategy. An important limitation was also pointed out in the paper. Further tests from the authors suggest that part of the predictable price changes that occurred during these 3- to 12-month holdingperiods may not be permanent, which can be later concluded as trend dissipation or a reverse trend. Such limitation is also discussed in "Short-Horizon Return Reversals and the Bid-Ask Spread", where the authors argued that most of the short-horizon return reversals can be explained by the way dealers set bid and ask prices, taking into account their inventory imbalances. The evidence indicated that the stock market may not be as liquid or resilient as has been generally believed [7]. These observations suggest if we fail to detect trend reversal when using the momentum strategy, it will be impossible for us to realize a significant profit and sustain the obtained profit in a relative long period.

Numerous papers have investigated the use of machine learning for financial time-series prediction and trading. Typically, the focus of the researchers concentrates on casting the underlying prediction problem as a standard regression or classification task[8, 9, 10], using regression models to forecast expected returns and classification models to predict the direction of future price movements. However, these experiments could lead to undesired performance in the context of time-series momentum for several reasons. First, sizing positions based on expected returns alone does not take risk characteristics into consideration, which could expose signals to large downside moves [3]. This is particularly important when it comes to raw momentum strategies without adequate risk adjustments, which are susceptible to large crashes during periods of market panic and depress. Second, the essence of automatic trading is a dynamic decision-making problem. It requires the researcher to explore an unknown environment and capture useful information in a timely manner. A major challenge lies in financial environment representation. The financial data contain a large amount of noise, jump, and movement, leading to the highly non-stationary time series[2]. Such issues can hardly be addressed by traditional machine learning approaches.

In light of the deficiencies of standard supervised learning techniques, new training methods and models are developed to account for trade-offs between risk and reward. In "Enhancing Time-Series Momentum Strategies Using Deep Neural Networks", Lim et al. introduces a novel class of hybrid models that combines deep learning-based trading signals with the volatility scaling framework used in time-series momentum strategies, which is referred to as deep momentum networks (DMNs)[4]. This improves existing momentum strategies for trading from several angles. First,

by using deep neural networks to directly generate trading signals, the authors remove the need to manually specify both the trend estimator and position sizing methodology, allowing them to be learned directly using modern time-series prediction architectures[4]. Second, by using automatic differentiation in existing backpropagation frameworks, they explicitly optimized networks for risk-adjusted performance metrics (the Sharpe ratio), improving the risk profile of the signal on the whole[1]. Similarly, Wood et al. added an online CPD module with Gaussian Processes to a DMN pipeline to improve overall returns of momentum strategy[1]. By incorporating the CPD module, they optimized their response to momentum turning points in a data-driven manner by passing outputs from the module as inputs to a DMN, which in turn learns trading rules and optimizes position based on some finance value function such as Sharpe Ratio[1]. With the help of the CPD module, the model learns how to exploit, but not overreact to noise at a shorter timescale. The strategy is able to exploit the fast reversion observed in DMNs but effectively balance this with a slow momentum strategy and improve returns across an entire Bull or Bear regime [1]. The idea is essential for momentum strategy. Intuitively, knowing "where to stop" is indispensable for a successful trading strategy. Although Wood et al. managed to achieve a considerable positive return by applying the strategy, the efficiency of the CPD module and the function of change point detection still have plenty of spaces for improvement with regard to automatic trading.

Compared to other learning-based approaches, reinforcement learning (RL) have less available literature yet its popularity is on the rise as more researchers begin to realize its significance in dealing with non-stationary time series[2, 11, 12, 13]. The first paper to implement deep reinforcement learning in designing a real trading system for financial signal representation and self-taught reinforcement trading is "Deep Direct Reinforcement Learning for Financial Signal Representation and Trading"[2]. The paper introduces a novel RDNN structure for simultaneous environment sensing and recurrent decision-making for online financial asset trading. The bulk of the RDNN is composed of two parts: DNN for feature learning and recurrent neural network (RNN) for RL. To further improve the robustness for market summarization, fuzzy learning concepts are introduced to reduce the uncertainty of the input data[2]. Another relevant research was done by Zhang et al., where they employed deep reinforcement learning algorithms to design trading strategies for continuous futures contracts[11]. Reward functions were improved by volatility scaling to scale trade positions based on market volatility. Methods were tested on 50 liquid futures contracts from 2011 to 2019, and their algorithms delivered positive profits despite

heavy transaction costs[11]. The two studies both offer comprehensive procedures for how state-of-the-art reinforcement learning was applied to improve the trading strategies. Nevertheless, momentum strategy with deep reinforcement learning has not been examined yet. It remains unclear how DRL will perform in momentum trading.

3. Solution

3.1. Problem Definition

We model the stock trading process as a Markov Decision Process (MDP). We then formulate our trading goal as a maximization problem. The algorithm is trained using Deep Reinforcement Learning (DRL) algorithms and the components of the reinforcement learning environment are:

* Action: The action space describes the allowed actions that the agent interacts with the environment. Normally, $a \in A$ includes three actions: $a \in \{-1, 0, 1\}$, where -1, 0, 1 represent selling, holding, and buying one stock. Also, an action can be carried upon multiple shares. We use an action space $\{-k, ..., -1, 0, 1, ..., k\}$, where k denotes the number of shares. For example, "Buy 10 shares of AAPL" or "Sell 10 shares of AAPL" are 10 or -10, respectively

* Reward Function (Momentum): $r(s_{p1}, s_{p2}, ..., s_{t-1}, a, s_t)$ is the incentive mechanism for an agent to learn a better momentum strategy (s_{pj} denotes a past state). The actual change of the portfolio value when action a is taken at state s_{t-1} and arriving at new state s_t is:

$$v_t - v_{t-1} = \sum_{i=1}^{n} R_t^i - R_{t-1}^i \tag{1}$$

$$R_t^i = X_t^i m_t^i r_t^i$$

X_t^i, m_t^i, r_t^i denotes the position size, volatility scaling and returns (per share) for stock i at period t.

To implement a momentum strategy, we adjust the reward function and blend the immediate change in portfolio value with a momentum, i.e., $r(s_{t-252}, s_{t-1}, a, s_t) = 0.5 \cdot (v_t - v_{t-1}) + 0.5 \cdot (v_t - v_{t-252})$, where v_t, v_{t-1} and v_{t-252} represent the portfolio values at state s_t, s_{t-1} and s_{t-252}[1] respectively. We capture the momentum by taking the monthly, quarterly and yearly returns into account and adjust the objective function accordingly. We assign different weights to the

[1]We assume 252, 63, 21 to be the total trading days within a year, a quarter, and a month.

momentum and more variations of the reward function can be found in the strategy section.The key idea here is that we maximize over the target 'momentum' rewards instead of focusing on the immediate change in portfolio values. The rewards may not necessarily be interpreted as the profit or gains of the trader.

* State: The state space describes the observations that the agent receives from the environment. Just as a human trader needs to analyze various information before executing a trade, so our trading agent observes many different features to better learn in an interactive environment.

* Environment: Multi-Stock Trading

The environment is based on FinRL and OpenAI Gym framework. We simulate live stock markets with real market data according to the principle of time-driven simulation

3.2. Data

The portfolio we select to study and train the DRL model is the SSE 50 Index constituents' stocks, which can be considered as a representative portfolio in the Chinese stock market[2]. The daily stock trading data featuring OHLC (open-high-low-close) was first retrieved from Yahoo Finance API by setting the attributes of the stock tickers of the stocks and the start and end date.

The corporate fundamental data used to calculate financial ratios was then located by their corresponding accounting principles from the financial statements of these constituent stocks. The authenticity of the data was endorsed by the Wind-Financial Terminal, to which the access was provided via the NYU Shanghai library service. The twenty collected features are Date, Quarterly operating income, Quarterly revenue, Quarterly net income, Assets, Shareholder's equity, EPS(Basic) incl. Extraordinary items, Common Equity, Common Shares Outstanding, Dividends per share, Current assets, Current liabilities, Cash & Equivalent, Receivables, Cost of Goods Sold, Inventories, Account payable, Long term debt, Debt in current liabilities, and Liabilities. Based on these numerical values retrieved from financial statements, ratios revealing the corporate information were calculated for later quantitative analytics. The financial ratios include Operating Margin, Net Profit Margin, Return on Assets, Return on Equity, Earnings Per Share, Book Per Share, Dividend Per Share, Liquidity ratios, Current ratio, Quick ratio, Cash ratio, Efficiency ratios, Inventory turnover ratio, Receivables turnover ratio, Payable turnover

[2]see Appendix A

ratio, Leverage financial ratios, Debt ratio, and Debt to Equity ratio. The calculated ratios will be used to set up the training environment and optimize the DDPG model.

The format of the raw data obtained from the Wind Terminal was different from Yahoo Finance, hence necessary reforming, cleaning, and manipulation were conducted to finally feed the data into the model. Besides, since the trading data are on daily basis and financial data are on quarterly bases, null values were removed in the ratio columns using backfilling after the two data frames were merged. The start date was selected on 2016-01-04 and the end date was selected on 2019-12-31 (1458 days and 1007 trading days). Based on these fundamental data, the financial ratios were then calculated to facilitate the quantitative analysis of the financial market in mainland China.

3.3. Model

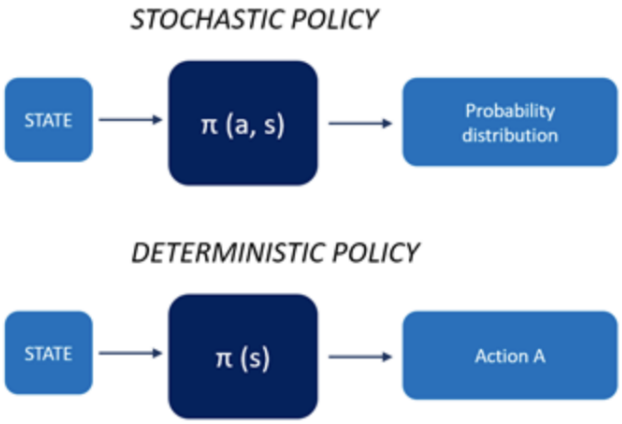

Figure 1: Stochastic Policy and Deterministic Policy

DDPG (Deep Deterministic Policy Gradient) is a policy learning method that incorporates deep learning neural networks into DPG (Deterministic Policy Gradient). For common stochastic strategies, the output of the neural network is the distribution of the actions. When determining each step of a series of actions, we need to get all the distributions of the strategy for sampling, and for some continuous actions with high dimensions, frequent sampling of the actions is computationally intensive and may lead to inefficiency and errors. In this case, the introduction of deterministic strategy could largely settles the issue. At the same state, the action is uniquely determined for a deterministic strategy[14] (see Figure1). Another limitation of standard rein-

forcement learning lies in the assumption of a discrete state space and actor space[15]. Methods widely applied to stochastic optimization such as Q-learning requires discretizing the continuous states and actions in advance, leading to the curse of dimensionality and increasing workload[14]. To deal with a trading environment continuous in both time and actions (as described in Problem Definition), algorithms such as PPO (Proximal Policy Optimization) and DDPG can be a perfect match for the environment.

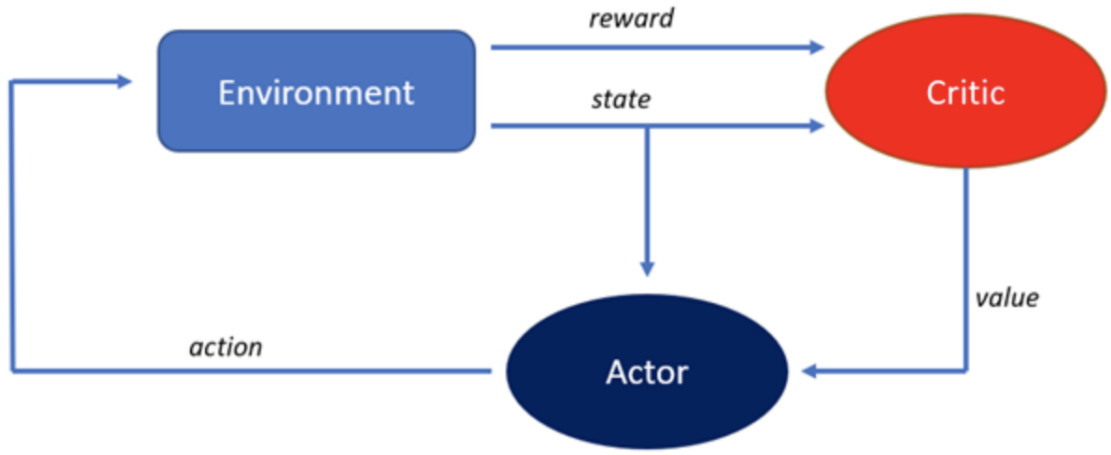

Figure 2: Critic Net and Actor Net

DDPG has two main networks: the critic net and the actor net (see Figure 2). The critic net is used to approximate the value function, where it receives action and state, and outputs a value of the state-action pair, i.e., $Q(s, a)$. Actor net, on the other hand, is used to approximate the policy function. Its input is state and its output is action value, i.e., $\tau(s)$. The flow between different networks is shown in Figure 3. The actor net receives s_t (state) from the environment. After the data flow, action a_t act on the environment to obtain rewards r_t, and then obtains s_{t+1} from the environment. DDPG will store the collection (s_t, a_t, r_t, s_{t+1}) in the experience replay buffer. After that, DDPG will randomly choose N collections in the buffer to construct a mini-batch and feed it into both the actor net and the critic net. With the mini-batch, the target net of the actor net generates a new action a_i and transfer it to the critic net. With the mini-batch and a_i, the target net of the critic net can calculate y_i and feed it into the online net. We can use θ^Q and θ^τ to denote the parameters of the two network respectively. In the DDPG algorithm, the critic net is updated by minimizing the following loss function[16]:

$$L(\theta^Q) = \frac{1}{R} \cdot \sum_i (y_i - Q(s_i, a_i|\theta^Q))^2. \tag{2}$$

where R denotes the memory capacity of the replay buffer and $y_i = r_i + \gamma Q'(s_{i+1}, \tau'(s_{i+1}|\theta^{\tau'})|\theta^Q)^2$ (γ is a discount factor). Meanwhile, we optimize the actor net by maximizing the policy objective function J [16]:

$$\nabla J(\theta^\tau) \approx \frac{1}{R} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\tau(s_i)} \nabla_{\theta^\tau} \tau(s|\theta^\tau)|_{s_i}. \tag{3}$$

The core improvements of DDPG over standard DPG algorithm and other actor-critic based reinforcement learning lie in the following two aspects. First, DDPG is particularly good at handling high-dimensional continuous action spaces[14] and thus a perfect match for multiple stocks trading, where we are dealing with continuous states and actions (changing asset, buying or selling shares of stocks). Second, compared with other applicable models such as PPO, DDPG has an advantage in dealing with non-stationary financial signals. As we discussed in the flow of the networks, the actor net of DDPG stores the transition data into the experience replay buffer (see Figure 3), and then randomly samples the mini-batch data from the experience replay buffer during training, so that the sampled data can be considered as uncorrelated[14]. This feature is particularly appealing to quantitative finance as non-stationary data has always been the major concern for traders and analysts.

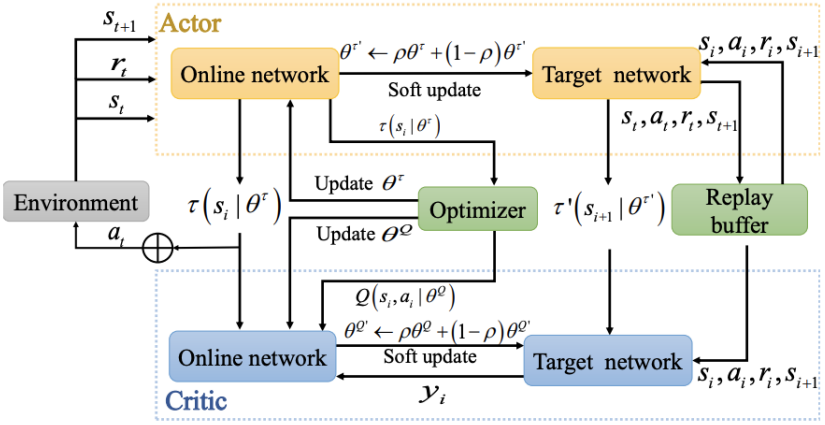

Figure 3: Architecture of DDPG[16]

Although most of the trials and back-testings in this paper are based on DDPG algorithm, we are also going to present our results under the PPO (Proximal Policy Optimization) model,

which has been widely used in deep reinforcement learning. PPO often serves as a baseline in literature with its strong stability and less variance. We are not going to discuss the details of this algorithm. Both the DDPG and PPO models are implemented using the packaged modules in FinRL library, based on Stable-Baseline3 of PyTorch and OpenAI Gym[17]. We first set up the corresponding trading environment and then initialize the agent of the models. Hyperparameters can be easily adjusted and tested with the powerful module of FinRL, largely increasing the efficiency of optimization of the training procedures.

3.4. Strategy

In this project, we select the *Buy&Hold* strategy as a benchmark. The agent buys stocks and holds them for a long period regardless of fluctuations in the market. This is a common and traditional strategy, widely discussed in literature. Another benchmark strategy is daily revenue optimization (denoted as D). It is the default strategy in FinRL and it performs pretty well in various trading environments since the intuition is straightforward, similar to a greedy algorithm. At every state, the agent maximizes its daily return. This strategy is problematic in the sense that it may lead to overfitting and can hardly be applied to different circumstances.

We are inspired by the research of Moskowitz[18] on time series momentum, where a basic momentum strategy can be described as

$$X_t^i = sgn(r_{t-252,t})$$

The strategy indicates that a decision is made purely based on annual return of the asset. To react quicker to momentum turning points, a faster signal , based on monthly returns was added[19], which gives the equation:

$$X_t^i = (1-w) \cdot sgn(r_{t-252,t}) + w \cdot sgn(r_{t-21,t})$$

where w represents the weight assigned to the yearly and monthly momentum. We capture the momentum of the market by adjusting the reward function of the DDPG network in a similar manner. The reward function can be defined as:

$$R = w_1 \cdot (v_t - v_{t-252}) + w_2 \cdot (v_t - v_{t-63}) + w_3 \cdot (v_t - v_{t-21}) + w_4 \cdot (v_t - v_{t-1})$$

$$\sum_j w_j = 1$$

We assign different weights to the yearly, quarterly, monthly and daily momentum (v_t denotes the total asset at period t). It will be complicated to quantify and measure the influence of this adjustments on the trading environment in DDPG. The basic idea is that the agent no longer follow the default greedy algorithm where only daily return at each stage is considered. Instead, it evaluates the momentum in the past periods and make decisions to optimize such evaluation. We believe by incorporating different momentum and signals, we will be able to achieve a higher risk-adjusted return and minimize the loss caused by trend reversal.

In addition, we also consider the log version and Sharp ratio as the reward functions. We take the logarithm form of the relative change in the value of portfolio to see how the performance of the strategy may vary. The Sharpe ratio serves as an important reference for us to measure whether our strategy indeed has the characteristics of achieving a risk-adjusted return and deal with trend reversal. The collection of all the strategies we explored is displayed in the following table 4.

Strategy	Yearly	Quarterly	Monthly	Daily
D[3]	0	0	0	1
Y	1	0	0	0
YQ	0.5	0.5	0	0
YQD	0.33	0.33	0	0.33
YQMD	0.25	0.25	0.25	0.25
log(Y)	1	0	0	0
log(YQ)	0.5	0.5	0	0
Sharp	0	0	0	0

Table 1: Trading Strategies and Related Weights

4. Results

We divide the data set into training and testing sets. The training set ranges from 2016-01-04 to 2019-01-04 (764 trading days) and the testing set ranges from 2016-01-04 to 2019-12-31 (244 trading days). Backtesting plays a key role in evaluating the performance of strategies. The Quantopian pyfolio package is used to backtest our trading strategies. It is easy to use and consists of various individual plots that provide a comprehensive image of the performance of a

[3] **Y, Q, M, D** denotes Yearly, Quarterly, Monthly and Daily momentum respectively.

trading strategy.

For hyperparameters tuning, we mainly focus on adjusting the buffer size, learning rate, batch size and the total timesteps of training. The buffer size represents the magnitude of the replay buffer used in DDPG and in order to generate stable behavior within the algorithm, the replay buffer should be large enough to contain a wide range of trading experiences. As we have discussed in the model section, the replay buffer plays a crucial role in dealing with the non-stationary characteristics of the data. We are less likely to suffer from the sampling of correlated elements and therefore the network becomes stable. Nevertheless, a large buffer requires enormous memory and it might decrease the efficiency of training. Learning rate and batch size are indispensable components in the optimization of the network. Choosing a proper learning rate will help us minimize the loss function properly and efficiently. Batch size is also used to increase the speed of optimization. Moreover, it helps deal with the potential overfitting of the mode. Total timesteps is the number of steps in total the agent will do for any environment. This parameter may be fixed given an environment with fixed steps. However, in our problem, this parameter is unknown as we cannot given an estimation of the total number of steps. All we can do is to do some experiments and play with this value. To adjust the hyperparameters, we further divide the training set into a smaller training set (2016-01-04 to 2018-01-04) and a validation set (2018-01-04 to 2019-01-04). After a series of experiments with different configurations[4], we decide to set buffer size = 50000, learning rate = 0.001, batch size = 128, total timesteps = 10000.

Strategy Performance						
Strategy	CR[5]	Volatility	Sharp	Calmar	Max Drawdown	Omega
$Buy\&Hold$	0.209	0.182	1.19	1.44	-0.153	1.24
D[6]	0.588	0.172	1.97	3.16	-0.121	1.51
Y	0.995	0.205	2.46	5.25	-0.118	1.68
YQ	0.668	0.184	2.04	3.20	-0.134	1.54
YQD	0.894	0.199	2.35	5.03	-0.112	1.64
$YQMD$	0.809	0.194	2.24	4.65	-0.110	1.60
$log(Y)$	0.972	0.208	2.39	4.89	-0.124	1.66
$log(YQ)$	0.838	0.189	2.35	4.84	-0.109	1.64
D_{PPO}	0.494	0.162	1.81	3.00	-0.108	1.46
Y_{PPO}	0.492	0.153	1.90	2.82	-0.114	1.51
YQ_{PPO}	0.429	0.148	1.76	2.27	-0.125	1.45
$YQMD_{PPO}$	0.439	0.137	1.93	2.51	-0.115	1.60
$Sharp$	0.735	0.195	2.07	3.24	-0.144	1.55

Table 2: Performance of Trading Strategies: Momentum and Benchmarks

[4]see Appendix A for detailed configurations

We train the model with our strategies described in the last section and we get the statistics showed in Table 2. Buy&Hold is the benchmark strategy. Daily optimization is the default strategy in FinRL which serves as another reference for the performance of our momentum strategy. We can see that among all the strategies being tested, the yearly momentum (Y) performs the best, with a cumulative return of 0.995 (ratio). It goes far beyond the benchmarks and also beat all the other momentum strategies. The volatility is relatively high but still, it has the highest Sharp, Calmar and Omega ratios, which suggests that it provides the highest risk-adjusted return as well. The Max Drawdown of the strategy is around the average among all strategies. It turns out that blending a shorter momentum does not improve the performance of the strategy. A possible explanation would be the overfitting of models given the short and detailed signals in the reward function. Nevertheless, the Max Drawdown of YQD and $YQMD$ is smaller than the yearly momentum. This observation indicates that blending a shorter momentum indeed has the function of detecting a reverse trend and could result in a shorter period of loss.

Taking the logarithm form of the relative change in the value of portfolios does not significantly improve the performance of our strategies. The cumulative returns slightly decrease for the yearly momentum, with higher volatility and smaller risk-adjusted return. For YQ, the condition is the opposite. All indicators of performance are improved. Still, we can hardly provide any interpretation to account for such observation due to inconsistency in the results. The momentum strategies trained by PPO perform worse than DDPG but beat the benchmark. Regardless of the diversity of strategies, the PPO model gives us similar results. The performance of these strategies features a low volatility and a small max drawdown, indicating a smooth and stable curve of returns. *Sharp* results in considerable returns, which is beyond our expectation. The strategy outperforms some of the momentum strategies and gives relatively high Sharp and Calmar ratio. The yearly momentum defeats the *Sharp* strategy and this shows that our momentum strategy is capable of achieving a high risk-adjusted return rather suffering from turbulence and trend dissipation.

[5] **CR**: Cumulative Returns
[6] **Y, Q, M, D** denotes Yearly, Quarterly, Monthly and Daily momentum respectively.

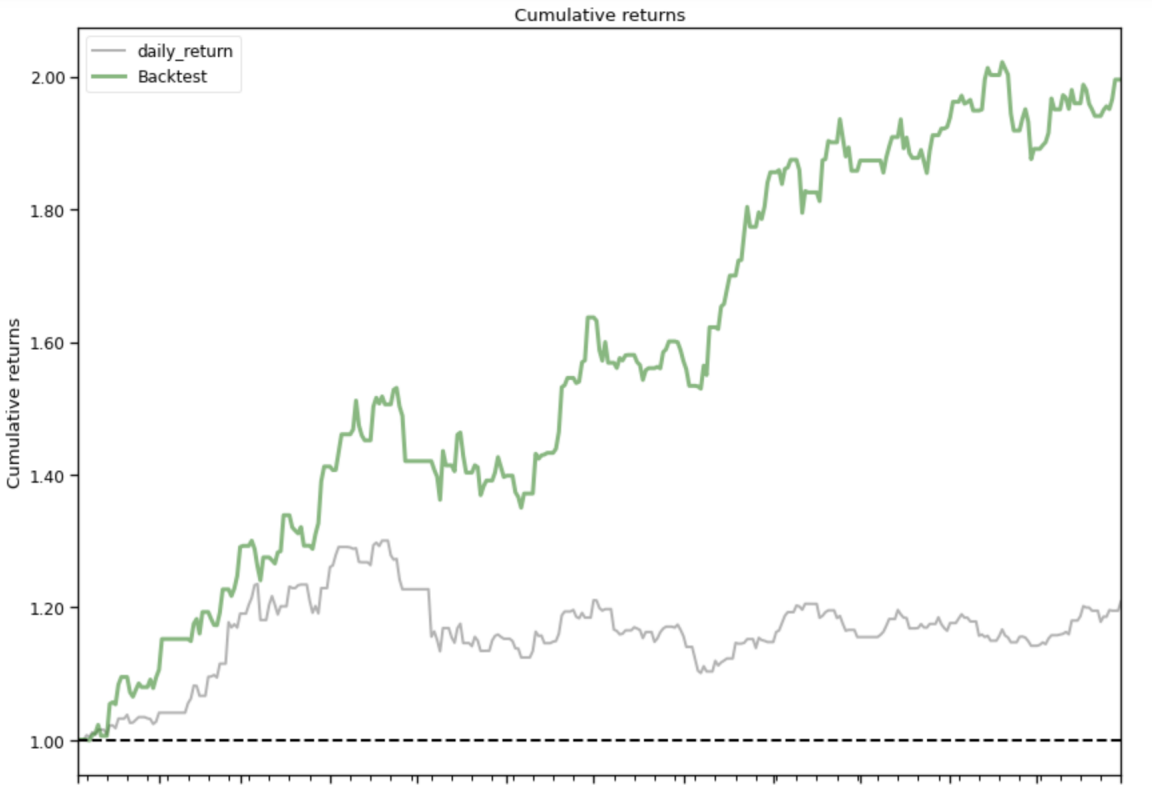

Figure 4: Cumulative Returns - Yearly Momentum

Figure 4 displays the cumulative return and daily return for Yearly Momentum (Y) strategy. We observe a nearly doubled cumulative returns and a smooth positive trend with a period of 12 months.

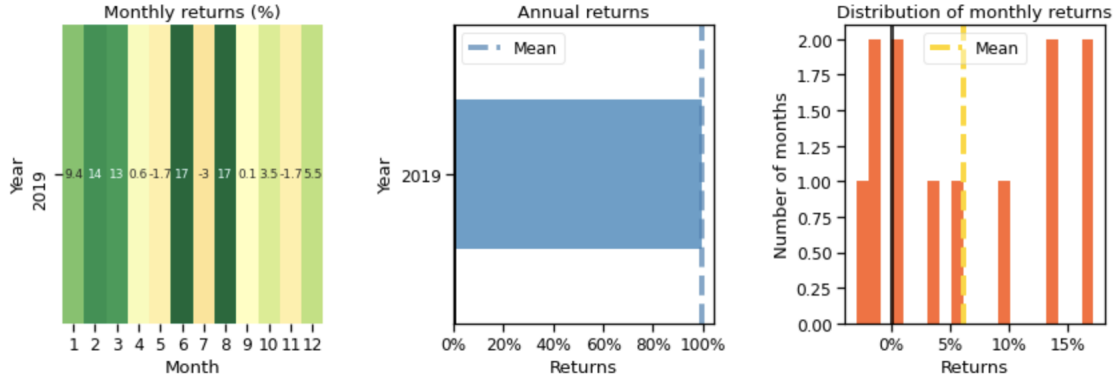

Figure 5: Return Decomposition

Figure 5 shows the detailed distributions of returns across the whole year, we find that June and August witness the highest monthly return in 2019. May, July and November witnessed a

negative return. For the whole year, most of the monthly returns are positive. On average, each month witnesses a return of around 6 percent.

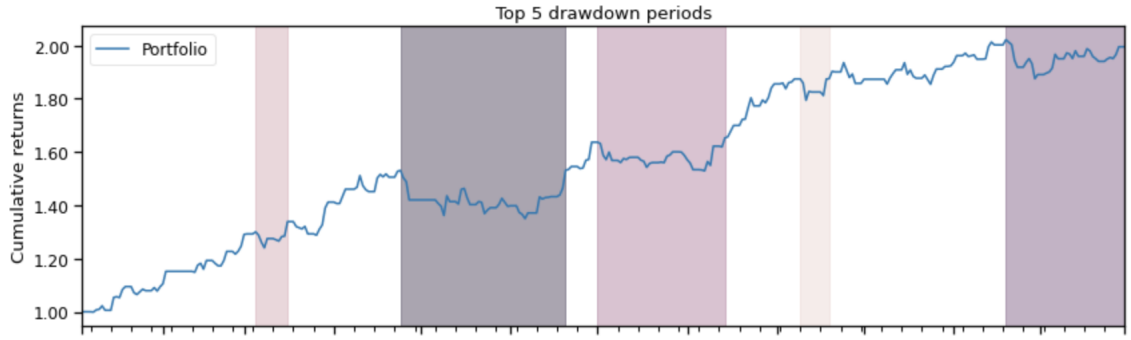

Figure 6: Top 5 Drawdown Period

As for the drawdowns, we see in Figure 6 that there are a few significant drawdown periods. Their lengths vary, but around the middle of the year there is a significant concentration of drawdown periods.

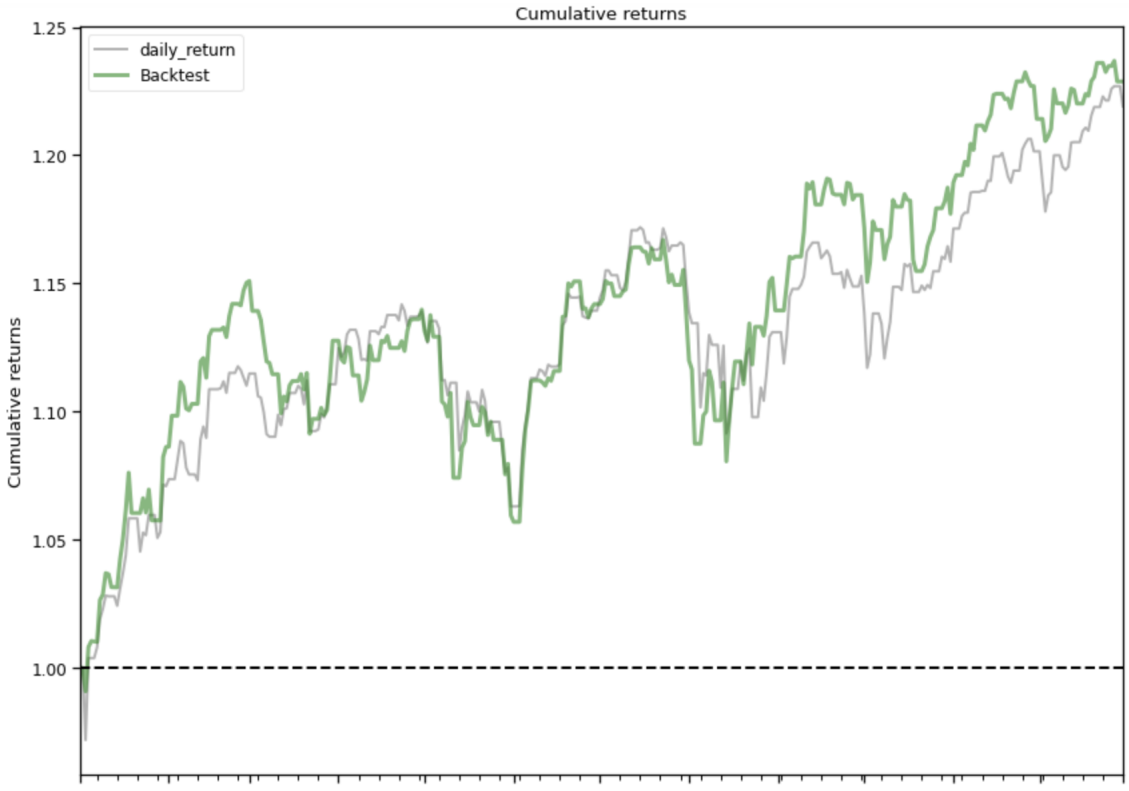

Figure 7: Cumulative Returns - Yearly Momentum for **Dows 30**

We also applied the same yearly momentum strategy to a different set of data , Dows 30

constituents stocks. The structure of the algorithm remains unchanged. We could consider it as a trial within the U.S stock market. The backtesting result shown in Figure 7 shows a similar pattern but a much smaller cumulative returns compared to the performance of yearly momentum in the Chinese market. Similar trials are implemented for other momentum strategies but the results are similar. The returns of the strategies barely defeat the buy&hold benchmark. It turns out the strategy does not function well in the U.S stock market.

5. Discussion

The limitations of our research lies in the following aspects. First of all, we chose a four-year period to train our model, which might be too short and could result in overfitting problems. Meanwhile, a larger portion of the data should be put into the training set to increase the validity of our model. Second, there are a larger number of empty entries within the fundamental data. These missing values may not cause significant issue regarding the trading environment but still, it may lead to some bias and inaccuracy. Extra research can be done to collect detailed information about individual stocks and fill in these entries.

A strong assumption in our model is that transaction is based on daily open and close prices. This setting is unrealistic in the real world as transaction can take place frequently during the same day (for instance, adjusting the positions). Besides, the daily losses and gains can hardly be modeled as a combination of differences between open and close price of the stocks. It is unlikely that an investor will routinely buy and sell stocks at the beginning/end of the day. In the future, we could extend our model and algorithm to intra-day trading and explore the possibility of adding smaller momentum within the day to our strategy.

Regarding the models, due to considerable amount of time devoted to exploring diverse strategies, we have less time left for adjusting the hyperparameters and we only try a limited set of configurations. We would like to spend more time finding the optimal hyperparameters if future studies are conducted. In addition, we would also like to explore other popular reinforcement learning models such as Long Short-term Memory (LSTM), which can automatically define the magnitude of the momentum.

6. Conclusion

Although not perfect, this is still our first attempt to study a momentum strategy using DRL. We did so since the exploration of Chinese stock market received less attention in the past. We compared the performances of different strategies in literature and tried to make improvements. Our strategy performs well in the Chinese stock market, beating benchmark strategy such as buy&hold, Daily optimization and strategies using PPO models.

Our strategy witnesses significant return, probably due to the bull market in China in 2019. SSE index and SSE 50 index achieves a increase of 20% and 40% respectively in 2019 and this market trend largely decides the ultimate return of our strategy. We could imagine a huge momentum persists in 2019 and therefore, the momentum strategies becomes a perfect match and works extremely well. This well explains why yearly momentum outperforms momentum strategies that blend a shorter momentum. The reverse trend, in this context, became a distraction and interrupt the agent from following the positive market trend. The "detected" reversals are mostly noises and lead to unnecessary drawdowns and a decrease in positions.

Our strategy produces ideal cumulative returns and Sharp/Calmar ratio, but features a large volatility and has larger drawdowns compared to strategies implemented by PPO. It remains unclear whether such volatility could lead to significant issues given a different context. Our strategies may not perform ideally given a different market trend, as we observed in the the U.S stock market. In fact, the western markets tend to have a mature framework, with abundant quantitative trading institutions and agents, closer to a zero-sum game compared with the environment in China where profit is largely decided by indexes and market trends. A more comprehensive comparison should be conducted to examine the distinctions between applying strategies in the Chinese and U.S market.

References

[1] K. Wood, S. Roberts, and S. Zohren, "Slow momentum with fast reversion: A trading strategy using deep learning and changepoint detection," *The Journal of Financial Data Science*, vol. 4, no. 1, p. 111–129, Dec 2021. [Online]. Available: http://dx.doi.org/10.3905/jfds.2021.1.081

[2] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep direct reinforcement learning for financial signal representation and trading," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 3, pp. 653–664, 2017.

[3] E. Saad, D. Prokhorov, and D. Wunsch, "Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks," *IEEE Transactions on Neural Networks*, vol. 9, no. 6, pp. 1456–1470, 1998.

[4] B. Lim, S. Zohren, and S. Roberts, "Enhancing time series momentum strategies using deep neural networks," *SSRN Electronic Journal*, 01 2019.

[5] N. Jegadeesh, "Evidence of predictable behavior of security returns," *The Journal of Finance*, vol. 45, no. 3, pp. 881–898, 1990. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1990.tb05110.x

[6] N. Jegadeesh and S. Titman, "Returns to Buying Winners and Selling Losers: Implications for Stock Market Efficiency," *Journal of Finance*, vol. 48, no. 1, pp. 65–91, March 1993. [Online]. Available: https://ideas.repec.org/a/bla/jfinan/v48y1993i1p65-91.html

[7] T. Jegadeesh, "Short-horizon return reversals and the bid-ask spread," *Journal of Financial Intermediation*, vol. 4, no. 2, pp. 116–132, 1995. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1042957385710066

[8] W. Bao, J. Yue, and Y. Rao, "A deep learning framework for financial time series using stacked autoencoders and long-short term memory," *PLoS ONE*, vol. 12, 2017.

[9] S. Gu, B. Kelly, and D. Xiu, "Empirical Asset Pricing via Machine Learning," *Review of Financial Studies*, vol. 33, no. 5, pp. 2223–2273, 2020. [Online]. Available: https://ideas.repec.org/a/oup/rfinst/v33y2020i5p2223-2273..html

[10] M. Binkowski, G. Marti, and P. Donnat, "Autoregressive convolutional neural networks for asynchronous time series," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 580–589. [Online]. Available: https://proceedings.mlr.press/v80/binkowski18a.html

[11] Z. Zhang, S. Zohren, and R. Stephen, "Deep reinforcement learning for trading." Institutional Investor Journals Umbrella, 2020. [Online]. Available: https://jfds.pm-research.com/content/early/2020/03/16/jfds.2020.1.030

[12] Z. Xiong, X. Liu, S. Zhong, H. Yang, and A. Walid, "Practical deep reinforcement learning approach for stock trading," *CoRR*, vol. abs/1811.07522, 2018. [Online]. Available: http://arxiv.org/abs/1811.07522

[13] T. G. Fischer, "Reinforcement learning in financial markets - a survey," Friedrich-Alexander University Erlangen-Nuremberg, Institute for Economics, FAU Discussion Papers in Economics 12/2018, 2018. [Online]. Available: https://ideas.repec.org/p/zbw/iwqwdp/122018.html

[14] C. Qiu, Y. Hu, Y. Chen, and B. Zeng, "Deep deterministic policy gradient (ddpg)-based energy harvesting wireless communications," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8577–8588, 2019.

[15] Y. Hou, L. Liu, Q. Wei, X. Xu, and C. Chen, "A novel ddpg method with prioritized experience replay," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2017, pp. 316–321.

[16] Y. Liu, Z. Jiang, S. Zhang, and S. Xu, "Deep reinforcement learning-based beam tracking for low-latency services in vehicular networks," 02 2020.

[17] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[18] T. Moskowitz, Y. Ooi, and L. Pedersen, "Time series momentum," *Journal of Financial Economics*, vol. 104, 09 2011.

[19] A. Garg, C. Goulding, C. Harvey, and M. Mazzoleni, "Momentum turning points," *SSRN Electronic Journal*, 01 2019.

A. Appendix

Table 3: Portfolio Details

No.	Index	Stock Name
1	600000	Shanghai Pudong Development Bank
2	600028	China Petroleum & Chemical Corporation
3	600030	CITIC Securities
4	600031	Sany Heavy Industries
5	600036	China Merchants Bank
6	600048	Poly Real Estate
7	600050	China United Network Communications
8	600104	SAIC Motor
9	600196	Shanghai Fosun Pharmaceutical
10	600276	Jiangsu Hengrui Pharmaceuticals
11	600309	Wanhua Chemical Group
12	600436	Zhangzhou Pientzehuang Pharmaceutical
13	600438	Tongwei Company
14	600519	Kweichow Moutai
15	600547	Shandong Gold Mining
16	600570	Hundsun Technologies
17	600585	Anhui Conch Cement
18	600588	Yonyou Network Technology
19	600690	Haier Smart Home
20	600745	Wingtech Technology
21	600809	Shanxi Xinghuacun Fen Wine Factory
22	600837	Haitong Securities
23	600887	Yili Group
24	600893	AECC Aviation Power
25	600900	China Yangtze Power
26	601012	Longi Green Energy Technology
27	601066	CSC Financial
28	601088	China Shenhua Energy
29	601138	Foxconn Industrial Internet
30	601166	Industrial Bank
31	601211	Guotai Junan Securities
32	601288	Agricultural Bank of China
33	601318	Ping An Insurance
34	601336	New China Life Insurance
35	601398	Industrial and Commercial Bank of China
36	601601	China Pacific Insurance
37	601628	China Life Insurance
38	601633	Great Wall Motor Company
39	601668	China State Construction Engineering
40	601688	Huatai Securities
41	601857	PetroChina
42	601888	China Tourism Group Duty Free Corporation
43	601899	Zijin Mining Group
44	601919	Cosco Shipping
45	603259	WuXi AppTec

Continued on next page

Table 3 – Portfolio Details

No.	**Index**	**Name of Stock**
46	603288	Foshan Haitian Flavouring & Food Co
47	603501	Will Semiconductor Company
48	603986	GigaDevice Semiconductor

Hyperparameters	Value
Buffer size	10000, 20000, 50000
Learning rate	0.0001, 0.001, 0.01
Batch size	32, 64, 128, 256
Total timesteps	5000, 10000, 25000, 50000

Table 4: Hyperparameters and Configurations