Proceedings of the 6th International Conference on Computing and Data Science
DOI: 10.54254/2755-2721/57/20241305

# Continuous trading strategy based on deep reinforcement learning

**Kailin Zhang**

School of Information Science and Technology, Guangdong University of Foreign Studies, Guangzhou 510006, China

kailinzhang2022@gmail.com

**Abstract.** Automatic trading policy has been researched with reinforcement learning (RL). Designing a profitable and applicable policy is of great significance for research in quantitative finance. Incoprating with deep learning, typical deep reinforcement learning (DRL) algorithms such as Proximal Policy Optimization (PPO) have shown their effectiveness. To improve the practical applicability, the manner in which we train our model should better simulate the dynamic of the stock market. A sliding window training strategy is a better solution, which employes training, validation and trading procedures on dataset with a sliding window. However, this empirical strategy can still be further investigated in algorithm evaluation and the experiment designation such as choice of sliding window. In this paper, we further investigated the continuous trading strategy (CTS). We evaluated the performance of a wider range of algorithms, including PPO, Deep Deterministic Policy Gradient (DDPG), Advantage Actor-Critic (A2C), Soft Actor-Critic (SAC). Moreover, we trained our models on a longer term period. We also provide detailed observations and suggestions on experiment settings. These discussions will facilitate researchers in their future work.

**Keywords:** Financial Technology, Quantitative Finance, Stock Trading, Reinforcement Learning, Deep Learning.

## 1. Introduction

The stocking trading policy is significant for individual investigators in the market. Generally, a policy is to help the investigator decide if they should buy, hold, or sell stocks at a given time. The ultimate goal of a policy is to maximize returns while minimizing the risk or costs. However, a profitable policy is often hard to design because of the intrinsic complex features of real markets.

In recent times, the integration of deep learning (DL) and reinforcement learning (RL) has gained considerable attention in the realm of automated stock trading strategies. Deep reinforcement learning (DRL) have offer a promising avenue for devising adaptive trading policies that leverage intricate patterns within stock trading data while aligning with our objectives concerning cost and returns.

To stimulate the dynamics of real stock markets, Xiong et al. [1] introduced a training technique with sliding windows. This approach orchestrates the commencement of training, validation, and trading at distinct timestamps, with the window's mobility facilitating iterative iterations of these stages. Nonetheless, this technique potentially encounters the issue of future data exposure, as the current test data could inadvertently contribute to future training and validation cycles due to window interweaving.

Addressing this limitation, an innovative training strategy [2] employed a continuous sliding window methodology to partition data and periodically train the dataset. This novel approach seamlessly amalgamated three windows, mitigating data leakage concerns. Significantly, this method was exclusively implemented with the Dow Jones dataset's Proximal Policy Optimization (PPO)[3] algorithm.

Building upon this foundation, our research endeavors to extend this investigation by evaluating the efficacy of alternative reinforcement learning algorithms, including Deep Deterministic Policy Gradient (DDPG)[4], PPO[3] and Advantage Actor-Critic (A2C)[5]. We incorporate data from the NASDAQ stock market to explore the effectiveness of other datasets. We aspire to analyze the intricate interplay between diverse parameters and features through experiments, thereby shedding light on the nuanced performance dynamics.

In Section 1, we outline the motivations of this research. Next, Section 2 focuses on recent works, discussing their focal points and contributions. Section 3 is to introduce the concept of continuous window training. In Section 4, we apply different experiments to analyze the performances of different algorithms, meticulously delineating our methodology and framework. Finally, Section 5 gives a overview on future implications.

## 2. Related Work

### *2.1. RL Algorithms Overview*

RL algorithms are widely applied in the Markov Decision Problem (MDP). An MDP consists of state s, action a, start state, one or more terminal states and a possible discount factor $\gamma$, a transition function $T(s,a,s')$ and a reward function $R(s,a,s')$, where s′ is the next state when performing $a$ in $s$. In these settings, a policy $\pi: S \to A$ is a function mapping the states to the action. We aim to find the optimal policy $\pi^*$, such that we can get the maximum expected reward.

The question with continuous space can be more complex. Our goal is to approximate the policy function, or policy optimization, which can work in an iterative manner. A policy iteration mainly consists of two processes: evaluation and improvement. In the evaluation process, we sample from the environment to estimate $V^\pi$ meanwhile minimize the estimated reward and the actual $V^\pi$. Then, the policy is updated with a policy gradient. When estimating the reward from sampling, we can apply the discount factor $\gamma$ to determine the influence of past samples to current estimation. A2C is the algorithm considering 2 previous steps from the current sample. Advantage estimation is a better choice since we can evaluate the performance of policy according to a baseline value. [6] prosed a generalized advantage estimation to consider a wider range of samples.

During the sampling process, each sample is collected based on the current policy, which can be good or bad. Trust Region Policy Optimization (TRPO) [7] designed a surrogate loss function, considering the compared ratio between the new and old policies when estimating the advantage. The difference of the policies is yield to the constraint defined by KL-divergence. PPO can further simplify This constrained optimization question by using the clipped surrogate loss.

In the Q-learning process, the DDPG first collects samples according to the current policy. Then, the Q-function is updated with the gradient. After that, the policy is updated by gradients computed from the Q-function. Considering the deterministic action, DDPG can be very sample-efficient but also unstable. On the other hand, the SAC applied the entropy of policy when estimating the Q-value. By using the soft version during the update process, the algorithm can have a better exploration and lower the risk of overfitting.

### *2.2. Trading with DRL*

DRL methods have shown great effectiveness in policy designation. Current researchers have used the historical-data-based manner to train an agent to validate and evaluate the performance of algorithms. Jiang et al.[8] proposed an actor-critic fashion reinforcement learning framework in portfolio management. With this inspiration, Liang et al. [9] focus on portfolio management with conventional

RL algorithms such as DDPG, PPO and PG. Guan et al. [10] further proposed an empirical approach with feature weights to explain the performance of DRL algorithms on portfolio management.

Zhang et al. [11] tested both continuous and discrete trading strategies in automatic stocking trading. Xiong et al.[1] proposed a sliding window training strategy to simulate the real stock market. They evaluated the performance of DDPG on the Dow Jones 30 dataset. This work is further explored in [12], where the researchers employed an ensemble strategy. In this research, we can change the predicting model based on the Sharpe ratio of the previous training period. This ensemble strategy includes A2C, PPO and DDPG. However, even though this is a better simulation of the real market, this method may have overfitting problem or looking ahead bias when the current validation window and the future testing window interweaved. To solve this problem, a better training strategy is proposed in [2]. The author integrated the three discrete windows together to avoid bias. The performance of PPO is better than the random forest and Dow Jones Index Average strategy.

However, this strategy is currently only evaluated in the short-term period. The author tested the model on 1 month of Dow Jones stock market data. Thus longer period of training should be considered. Meanwhile, the remaining algorithms such as DDPG and A2C are not being evaluated based on this training strategy. Detailed comparisons of different DRL algorithms can also facilitate the choice of different policies. Since the current dataset mostly focused on the Dow Jones stock market, other stock markets, such as NASDAQ with more companies, are also waiting to be explored.

Considering all the questions mentioned above, we aim to evaluate the performance of different DRL algorithms on the NASDAQ stock market trading data. Specifically, we will apply the longer-term continuous window strategy to train, validate, and trade with models. With different comparison experiments, we wish to explain the effectiveness or shortages of different algorithms on NASDAQ stock data. This work can benefit researchers who wish to design more profitable trading policies.

## 3. Question Description

A typical reinforcement learning algorithm includes an agent, environment, action, policy, loss, and reward. In the inspiration of Liu et al. [1], we formulate our question as follows. Given an individual investigator who wants to investigate the NASDAQ stock market, if the investigator has 1, 000, 000 dollar initial asset, how can the investigator design a policy to maximize the expected return? We suppose that an investigator can buy, sell, or hold a company's share with an exact amount, where the maximum amount of buying/sell is 100 and the minimum is 0. We denote buy with positive sign and sell as negative sign, and zero represents hold. Therefore, the amount of company $j$ on day $i$ can be $-100 \leq a_{ij} \leq 100$. We define our environment on timestep $i$ as $s_i = \{r,\ \boldsymbol{p},\ \boldsymbol{v},\ \boldsymbol{f}\}$, where $r$ represents the remaining asset, $\boldsymbol{p} \in \mathbf{R}^{90}$ is the close price, $\boldsymbol{v} \in \mathbf{R}^{90}$ is the volume of all companies, meanwhile $\boldsymbol{f} \in \mathbf{R}^{360}$ denotes the additional 4 technical indictor features. Moreover, the policy $\pi$ is a function mapping from environment to action. The reward $r(s, a, s')$ is the return of taking action a from state $s$ to $s'$.

## 4. Experiment

### 4.1. Data Preprocessing

We used the stock market data from 90 companies in the NASDAQ stock market. The benchmark is the NDX index, the average of the NASDAQ stock market. The timeline is selected from 2023/01/10 to 2023/05/23. There are 90 trading days in this period. For each day, our original data contains the open price, close price, high price, low price, volume, and the ticker of each company. Technical indicators are widely used in the feature engineering of time series. Following works from [2], [10], we added technical indicators as follows:

1) Moving Average Convergence Divergence (MACD): MACD is a powerful momentum indicator that highlights trends and potential reversals by comparing short-term and long-term moving averages of an asset's price. It is instrumental in identifying bullish and bearish signals.

2) Relative Strength Index (RSI): RSI is a widely utilized oscillator that quantifies the magnitude and velocity of recent price changes. It provides insights into overbought and oversold conditions, aiding in the assessment of potential trend reversals.

3) Commodity Channel Index (CCI): CCI is a versatile indicator that helps identify the cyclical nature of stock prices. It measures an asset's price deviation from its statistical average, enabling the detection of potential trend shifts.

4) Directional Movement Index (DX): DX is a component of the Average Directional Index (ADX) and is used to evaluate the strength and direction of a trend. It assists in determining whether a trend is developing or weakening.

## *4.2. Trading Simulation*

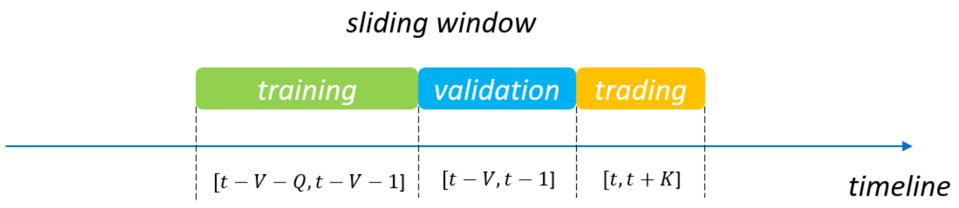

**Figure 1.** Sliding Window of Continuous Training Strategy.

Most researchers divide the time series dataset into in-sample and out-of-sample data. The start of training, validation, and trading is pre-defined. To better simulate the stock market's dynamic nature while avoiding the look-ahead bias, we follow the pipeline of [2], which we call this continuous training strategy (CTS) as shown in figure 1. In CTS, the validation and trading data continuously follow the training process. The CTS pipeline is a improved version of discrete training strategy (DTS), where consists three individual sliding windows. The three data sections are discrete from each other.

As shown in figure 1, In this pipeline, we design a sliding window moving from the beginning to the end of the timeline. A window consists of three sections: training, validation, and trading. If a window is in the length of $W$, we can use first $Q$ days for training, $V$ days for validation, and the remaining K days for trading. For every timestep t, the training data is from day $t-V-Q$ to day $t-V-1$. We then use data from day $t-V$ to $t-1$ for validation. The model is then retrained in the training and validation section. Next, we apply the trading strategy from day $t$ to day $t+K-1$. A base case is $K=1$, meaning we only trade on 1 day for every timestep. However, the base case training strategy may not be efficient enough since the sliding window can move toward only 1 day for each timestep. This is not recommendable for the longer-term data, which mostly consists of several months or years.

To explore the effectiveness of the trading pipeline, we set the training window, validation window, and trading window to 24, 9, 3, respectively. We apply the training strategy to 5 months of data. Issues of this training pipeline are further discussed in Section 5.

## *4.3. Evaluation*

When designing the experiment, we mainly focus on three aspects: the technical performance of algorithms, the applicability in practical scenarios, and the comparison with the DTS. When evaluating the technical performance, we mainly tune the parameter of DDPG. We then applied several algorithms to analyze the applicability and effectiveness of the previous training strategy.

*4.3.1. Technical Performance*. To begin with, we provide oberservations of several training indicators when applying our training pipeline.

1) Loss: Our experiments found that the actor loss converges quickly, while the critic loss converges more slowly. This indicates that the actor-network is so confident in the policy, which limits the

performance of the critic network. Moreover, we observed that the critic loss can be unstable in some situations. A better performance is the critic loss can jump to a peak every time we train on a new sliding window, then converge over time steps.

2) Reward: The reward of each algorithm converges quicker than the loss. We have tried to increase the sample size or apply noise when choosing actions in DDPG, but these methods only increase to a small extent.

3) Learning Rate: The learning rate determines how fast the algorithms learn in each timestep. We observed that the learning rate is important for the stability of critic loss, thereby influencing the algorithm's performance. We found that a good setting of learning rate can. For example, as shown in figure 2, we compared the performance of DDPG using different learning rates, with 0.0001, 0.0005, and 0.001, respectively. When we set the learning rate to 0.0001, the cumulative return is the highest of the three settings.

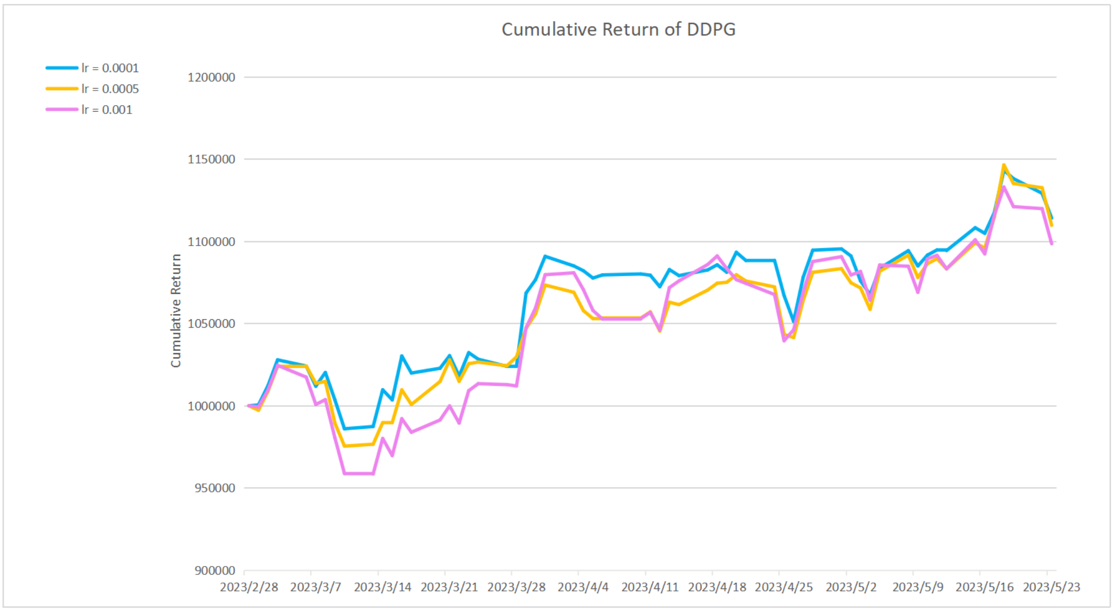

**Figure 2.** Cumulative Return of DDPG with Different Learning Rates.

4) Timesteps: The number of timesteps is related to the algorithm's update. Moreover, even if increasing the timestep allows the algorithm to converge better in loss, a better convergence probably does not guarantee better applicability. We compared the performance of DDPG over different time steps, with 2, 000 and 10, 000, respectively. Although the algorithm converges better when we set timesteps to 10, 000, the cumulative return is not as high as when we update for only 2, 000 timesteps.

5) Other parameters and indicators: Other parameters and indicators are worth noticing when training the model, such as KL-divergence, which determines the difference between the new and old policies.

*4.3.2. Applicability.* A good trading policy should be applicable in practical scenarios and facilitate the decision of individual investigators. We evaluated the applicability of different algorithms, including DDPG, PPO, A2C, and SAC. Following the standard of Liu et al., the benchmark of our experiment is the expected return with a close price of the NDX index. Firstly, we compared the cumulative return, which is an indicator of profitability.

**Table 1.** Performance Comparison between Different Algorithms and NDX

| | NDX | DDPG | PPO | A2C | SAC |
|---|---|---|---|---|---|
| Annual Return | **0.817** | **0.549** | 0.165 | 0.117 | 0.318 |
| Cumulative return | **0.150** | **0.110** | 0.037 | 0.027 | 0.068 |
| Max. drawdown | **0.038** | **0.048** | 0.064 | 0.054 | 0.058 |

Secondly, as shown in table 1, we provide our experiment results of indicators with different algorithms.

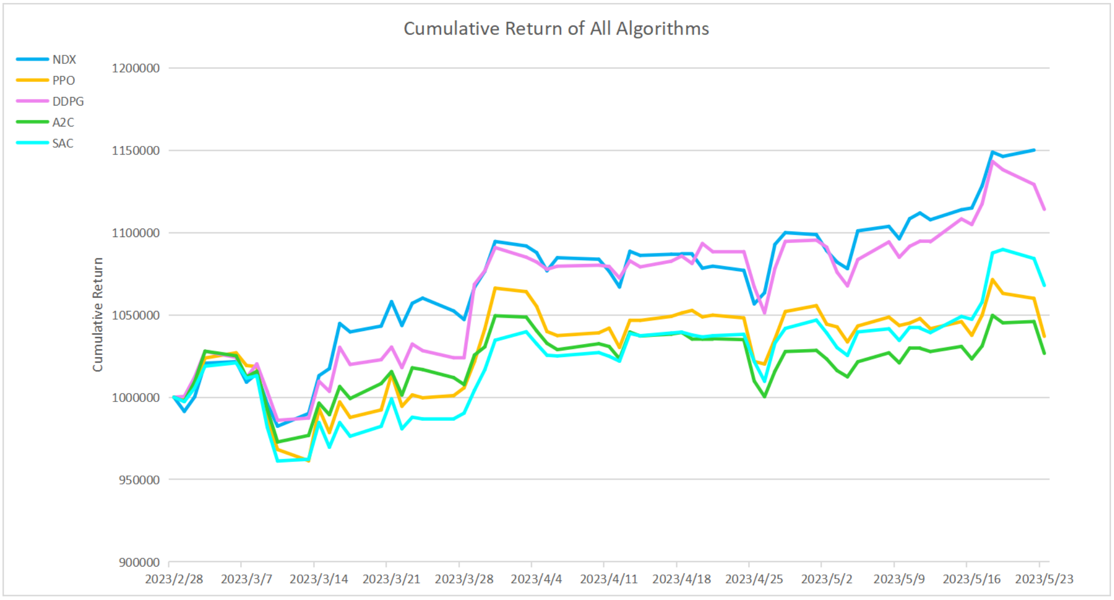

**Figure 3.** Cumulative Return of All Algorithms.

When focusing on profitability, we found that the DDPG algorithm gets the highest cumulative return of the four algorithms. As shown in figure 3 and table 1, the SAC gets a better return than the A2C, possibly because of the soft version of the reward function. However, although PPO is a popular algorithm that many investigators widely apply, this algorithm does not achieve a better profit than our baseline.

The other indicators are evident to explore the effectiveness of different algorithms. For example, DDPG achieves the highest stability of all the algorithms. Meanwhile, the highest DDPG also indicates the excellent performance of the trading policy.

*4.3.3. Effectiveness of Continuous Training Pipeline*. Since the CTS training pipeline is empirically designed, we further provide some analysis that are worth noticing.

1) Size of the sliding window: The CTS setting is more data efficient than the DTP. The CTS allows re-using current validation and test data for further training. However, the DTS requires a longer time range to train with the same amount of data.

2) Better Capture of Data Dynamics: The re-usage of current data for further training also helps the algorithm better capture stock data dynamics.

3) Usage of Trading Data: By applying the CTS, the trading for each set of future data is limited to only once, which addresses the look-ahead bias of the DTS.

## 5. Future Work

In Section IV, we provided observations and analysis of our experiments. Considering the novelty of this training strategy and the experience in our research, we further share some considerations with researchers.

1) Data Preprocessing: We added several commonly used technical indicators when processing the data. However, the training pipeline is based on sliding windows, which contain only information in a short period. Even though higher frequency data can be more representative, it may have the problem of dimensionality. Therefore, better feature engineering, such as using neural networks, can capture the inner pattern of time series.

2) Training Pipeline: In our training settings, we applied a wider sliding window, which is beneficial for longer-term training. However, the width of each part in the sliding window can probably affect the performance of training, validation, and trading.

3) Model Performance: We have evaluated the performance of several algorithms. However, most models are less effective than using the NDX index for reference. This issue can probably be addressed by better feature engineering. Additionally, we observed that the model's performance is sensitive to hyper-parameter settings. For example, the instability of DDPG affects the replicability of the experiment. Therefore, a more detailed algorithm analysis can be performed in future work.

## 6. Conclusion

This research mainly focuses on the policy designation of trading assisted by reinforcement learning agents. Even if the continuous training strategy can better capture the dynamic of stock market, the choice of sliding window, as well as the technical performance, profitability, applicability of algorithms are still worth for further research. We invstigated the sliding window training strategy on a longer-term timeline. We evaluated a wider range of algorithms with different parameters, including DDPG, PPO, A2C and SAC. By comparing with NDX benchmark, we conluded that DDPG achieves the best performancem. However, it's worth noting that DDPG's performance was found to be sensitive to hyperparameters, indicating the need for careful tuning and optimization when implementing this algorithm in real-world trading scenarios. After that, we shared some considerations from our practical experience. For example, researchers can optimize the performance of algorithms by analyzing loss, reward, learning rate, timesteps and KL-divergence. Moreover, we provide advices on data processing, designation of training pipeline and how to apply algorithms. The use of neural networks, for instance, can help capture intricate temporal patterns, addressing the issue of dimensionality associated with higher-frequency data. All these advices can benefit future research on this training pipeline.

## References

[1] Xiong Z, Liu X-Y, Zhong S, Yang H, Elwalid A. Practical Deep Reinforcement Learning Approach for Stock Trading. arXiv preprint arXiv:181107522. 2018:1-7.
[2] Liu X-Y, Xia Z. Empirical Demonstration of Stock Paper Trading for Financial Reinforcement Learning. Available at SSRN 4253133. 2022.
[3] Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. arXiv preprint arXiv:170706347. 2017.
[4] Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, et al. Continuous control with deep reinforcement learning. arXiv preprint arXiv:150902971. 2015.
[5] Mnih V, Badia AP, Mirza M, Graves A, Lillicrap T, Harley T, et al., editors. Asynchronous methods for deep reinforcement learning. International conference on machine learning; 2016: PMLR.
[6] Schulman J, Moritz P, Levine S, Jordan M, Abbeel P. High-dimensional continuous control using generalized advantage estimation. arXiv preprint arXiv:150602438. 2015.
[7] Schulman J, Levine S, Abbeel P, Jordan M, Moritz P, editors. Trust region policy optimization. International conference on machine learning; 2015: PMLR.

[8] Jiang Z, Xu D, Liang J. A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem. 2017.
[9] Liang Z, Chen H, Zhu J, Jiang K, Li Y. Adversarial Deep Reinforcement Learning in Portfolio Management. arXiv preprint arXiv:180809940. 2018.
[10] Guan M, Liu X-Y. Explainable deep reinforcement learning for portfolio management. Proceedings of the Second ACM International Conference on AI in Finance2021. p. 1-9.
[11] Zhang Z, Zohren S, Stephen R. Deep reinforcement learning for trading. The Journal of Financial Data Science. 2020.
[12] Yang H, Liu X-Y, Zhong S, Walid A, editors. Deep reinforcement learning for automated stock trading: An ensemble strategy. Proceedings of the first ACM international conference on AI in finance; 2020.