

# Automating Stock Trading with Reinforcement Learning

**Team:** Yicheng Shen, Chu'An Li, Dingyuan Liu. **Project Mentor TA:** Jason Ma.

## 1) Abstract

For this project, we study the problem of automated stock trading using reinforcement learning, which is important for individuals and financial institutions that aim to maximize returns of their stock portfolio. We leverage a deep reinforcement learning (DRL) library in quantitative finance called FinRL to simulate the environment of a stock market. Our contributions include the acquisition of a new dataset and feature engineering, as well as performance analyses over different trading environmental attributes. We preprocess our new dataset and verify its validity through experiments. Moreover, we find parameters, such as the turbulence threshold and the transaction cost percentage, have explainable influences on overall performance. Studying the effect of these parameters provides insights for not only the training of an automated stock trading agent, but also real-world trading scenarios.

## 2) Introduction

**Set up the problem:** Our task is to maximize the cumulative return of a virtual portfolio containing a number of stocks. We focus on 30 stocks in the Dow Jones Industrial Average (DJIA). We use reinforcement learning (RL) to train an automated trading agent on historical stock and financial data. We leverage the existing FinRL framework to train the agent [Xiao-Yang Liu, 2021].

Essentially, the problem we aim to solve with RL can be described as a Markov Decision Process (MDP) [Leslie Kaelbling, 1996]. The optimization objective is to maximize the cumulative expected return. In our problem formulation, the main components are the following:

**Environment:** The environment consists of stock and financial data of 30 stocks in the DJIA along with hyperparameters that simulate a real market.

**State space  $\mathcal{S}$ :** The state space represents all the information an agent can observe in the market environment. Each  $s \in \mathcal{S}$  represents a specific state in the environment. In our setup, every state  $s$  contains historical stock data and financial data on a specific date.

**Action space  $\mathcal{A}$ :** The action space consists of all actions that an agent can perform. In our setup, each action  $a \in \mathcal{A}$  represents a set of trading decisions which includes selling, holding, and buying a number of shares of a stock or multiple stocks.

Example: Suppose we only consider one stock AAPL. We hold 100 shares of it and there are  $n$  shares on the market. In this case, the action space  $\mathcal{A}$  is  $\{-100, \dots, -1, 0, 1, \dots, n\}$ . An action  $a \in \mathcal{A}$  could be selling at most all 100 shares ( $a = i, i \geq -100$ ), holding the current shares ( $a = 0$ ), or buying a number of shares ( $a = i, i \leq n$ ).

**Reward function  $\mathcal{R}$ :** The reward function  $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$  maps a tuple  $(s, a)$  consisted of a state and an action to a reward  $r$  which is a real value. It incentivizes the agent to learn a policy that generates a higher return. In our setup, the reward at time  $t$  is the difference between the return in state  $s_t$  and the return in state  $s_{t+1}$  after the action  $a_t$  is taken.

At time  $t$ , the objective that we want to maximize can be written as  $E(\sum_{t=0}^{\infty} \gamma^t r_t)$  where  $r_t = \mathcal{R}(s_t, a_t)$  and  $\gamma$  is the discount factor ( $0 \leq \gamma \leq 1$ ).

To evaluate the agent's performance, we will use the cumulative return compared with the market index as our major performance metric.

**Motivation:** The approach to training the automated stock trading agent is not restricted to our virtual stock portfolio. Potentially, it could be scaled up and perform decision-making in the real market. Our project could also show an analysis of different choices of simulated factors to provide investors with explainability and some insights into realistic scenarios. Moreover, if this project succeeds in maximizing the value of a virtual portfolio with a performance better than manual methods, it would enhance the investment capability of both individuals and financial institutions.

### 3) Background

The major prior work that is related to our project is FinRL, which is a deep reinforcement learning (DRL) library that focuses on stock trading. It consists of three layers: i) an environment layer that simulates the financial market constructed with historic data, ii) an agent layer that provides DRL algorithms, and iii) an application layer for automated stock trading. This framework serves as the backbone of our project to train an automated stock trading agent. As pointed out by Xiao-Yang who is one of the main contributors to FinRL, marketing environments, especially historical data API, engage essential interaction with agents for trading tasks. Tutorial Stock\_Fundamental.ipynb provided by Xiao-Yang shows how FinRL facilitates developers to expose themselves to quantitative finance and to develop their stock trading strategies. Nevertheless, there are potential improvements available in terms of incorporating more relevant financial ratios as evaluation standards for multiple stock tradings. The notebook provides a good starting point for us: it contains a basic pipeline of using FinRL to train a trading agent and develop a good trading policy.

- A. Xiao-Yang Liu and Hongyang Yang and Jiechao Gao et al, "FinRL: Deep Reinforcement Learning Framework to Automate Trading in Quantitative Finance", ICAIF 2021. Github implementation at: <https://github.com/AI4Finance-Foundation/FinRL>.
- B. Stock\_Fundamental.ipynb. [https://github.com/AI4Finance-Foundation/FinRL/blob/master/tutorials/1-Introduction/Stock\\_Fundamental.ipynb](https://github.com/AI4Finance-Foundation/FinRL/blob/master/tutorials/1-Introduction/Stock_Fundamental.ipynb).

### 4) Summary of Our Contributions

1. **Contribution(s) in Application/Data:** We add new data for training our trading agent. We collected historical financial data from Wharton Research Data Services (WRDS). We use the new data along with historical stock data to set up the market environment.
2. **Contribution(s) in Feature Engineering:** We transform stock and financial data into useful ratios and use them as features in training. The additional features augment the state space and assist our trading agent's decision making.
3. **Contribution(s) in Analysis:** We analyze the agent's performance in terms of cumulative return in environments with different training datasets, turbulence thresholds, and

transaction cost percentages, in an explainable manner. The contribution in data is verified in this section.

## 5) Detailed Description of Contributions

### 5.0 Detailed Description of Each Contribution:

**Contributions in Application/Data:** We leverage Wharton Research Data Services (WRDS) and search for financial data to complement the stock data from Yahoo Finance. WRDS is a business data research service from The Wharton School providing data access for academic research purposes. We find the Financial Ratios Firm Level database to be a good fit for our project. We query the database by the DJIA stock tickers in the date range of 2009-01-01 to 2022-10-31. We obtain monthly financial data (containing null values) that indicate companies' valuation, profitability, liquidity, etc. The new data contains 4817 rows and 78 columns which contain financial ratios, such as net profit margin, return on equity, etc. Several columns contain roughly 15% of null values. The challenge is to figure out the right condition to drop a column. Note that since we are dealing with stock and financial data which are time series, we cannot simply drop rows with null values. Initially, we set the tolerance for null values to be 1% and we have 40 columns of financial ratios left. For the very few null values left, we fill them with 0. As we try to improve the trading agent's performance, we increase the tolerance to 10% to have more features (16 more columns) and larger state space. Another decision we made with the data is to average monthly data into quarterly data. This manipulation of the data aligns with real-world timing because companies' financial statements are usually released quarterly. This also helps to solve the issue of null values because some null values could be filled in by the average values. After all the preprocessing, we merge the new financial data with the stock data. This forms our new dataset that is ready for feature engineering and setting up the environment.

**Contributions in Feature Engineering:** We explore various ways to augment the state space. One effective way we found is through feature engineering. We perform feature engineering on the stock data to get a number of technical indicators, such as moving average convergence divergence (MCAD), Bollinger bands, relative strength index (RSI), etc. Moreover, we also take advantage of financial ratios created with the use of numerical values taken from financial statements to gain company-specific information. Among all the financial ratios, we have 5 categories: liquidity ratios, leverage ratios, efficiency ratios, profitability ratios, and market value ratios. For each category, there are some common and important ratios. Take Liquidity ratios as example: it contains current ratio, cash ratio, operating cash flow ratio, etc. Some ratios stem from others, resulting in correlations among some ratios. Thus, we try to include more ratios while making sure the involved ratios are important. The first step is to select one centroid of the five clusters using finance interpretation. For instance, debt ratio would always be included in leverage ratios. Then we set the rest of the ratios as features of PCA. To determine an appropriate  $n\_components$ , we plot the cumulative sum of explained variance ratios and also a line for 0.95 variance (plot attached at the supplement information section). When  $n$  components were around 20, the explained variance ratios started to plateau.

**Contributions in Analysis:** The FinRL tutorial notebook sets up a single environment containing a fixed turbulence threshold and a fixed transaction cost. The threshold controls

agent risk aversion and the transaction cost simulates a cost incurred in the real market. Fixing these factors limits the total amount of agent's operations and such lack of environmental variety conceals the potential of RL agents. We study a more complicated market environment that is less ideal, but we expect our agents to still perform better than the baseline. We supplement the previous work by tuning turbulence thresholds based on closed price turbulence distribution over the stock dataset, in order to simulate environments of different degrees of volatility. We also adjust the transaction cost according to the real-world situation, which usually presents as a commission fee equal to 1-2% of a total amount of buy or sell. We compare the performance under a turbulence threshold space of 3 and a transaction cost space of 3.

## 5.1 Methods

In this section, we put our contributions in a pipeline and describe our approach in a coherent manner. First, We query the Financial Ratios Firm Level database from WRDS and obtain financial data. We preprocess the acquired data and merge with the DJIA stock data. We perform feature engineering to obtain the stock technical indicators and financial ratios that we supply to the training of our agents. Then, we split our new dataset: training data spans from 2009-01-01 to 2019-01-01; test data spans from 2020-08-01 to 2021-9-30.

We use the FinRL framework to train our stock trading agent. We integrate the new dataset into the general trading environment in FinRL. By leveraging a DRL algorithm, we can encourage exploration and exploitation behaviors at the appropriate timing. In the dynamic environment that we set up with historical stock data and financial data, the trading agent improves its decision making about the choice of stocks, the choice of trading or holding on, the price to trade, and the quantity to trade. In more details, at the end of day of given time  $t$ , we will know the open-high-low-close price of the DJIA constituents stocks, the financial data, and other features we generated. We refer those data as state  $s$ . Then, according to the state space, the trader will output a list of actions, treating this value as the trading signal to buy or sell [Hongyang Yang, 2011]. Actions to trade would be based on DRL suggestions, expecting the cumulative shares would grow at the end of day. We then add the shares to buy to shares previously held, while subtract shares to sell to get updated shares held. Then at time  $t+1$ , following the same process, we can calculate the reward. The reward at time  $t$  is the difference between the return in state  $s_t$  and the return in state  $s_{t+1}$  after the action  $a_t$  is taken. We learn from the positive reward. The whole procedure would repeat until termination criteria reached.

Once we have the pipeline for training and testing an agent running, we study the effects of different *turbulence thresholds* and *transaction costs*. A turbulence threshold in a given environment is a user-defined scalar which controls the agent's buy and sell actions. The turbulence index is a measurement of the extent of volatility, or the (un)confidence that the stock's price would be relatively stable, of a specific stock, on a specific day. This index is calculated by the preprocessor of the FinRL framework, merely given a series of closed prices. In any given state, the agent extracts turbulence indexes from every stock's data and compares them with the turbulence threshold to determine which of the stocks are volatile. If true, the agent would avert risky operations. For example, the agent would not buy any shares of a certain stock if it senses its high volatility regardless of current price. In our new dataset, 90% of

stocks have turbulence less than 100 at any time, and 99% of stocks have turbulence less than 400 at any time. Therefore we choose 100 and 400 as our turbulence thresholds for testing. Note that a turbulence threshold of None means the same as an infinitely large threshold.

A transaction cost is also a user-defined scalar representing the fee paid for each buy or sell operation. It is deducted percentage-wise from the reward function in every non-hold action. The change in transaction cost may influence the agent’s strategy in terms of stock selection and operation period. The FinRL framework uses a low percentage of 0.1% in tutorials and other applications. However, 1-2% is a common range of transaction cost percentage among real-world brokers. We choose 0.01 and 0.02 as our transaction costs to better simulate a real market and evaluate the effect of these difference choices.

## 5.2 Experiments and Results

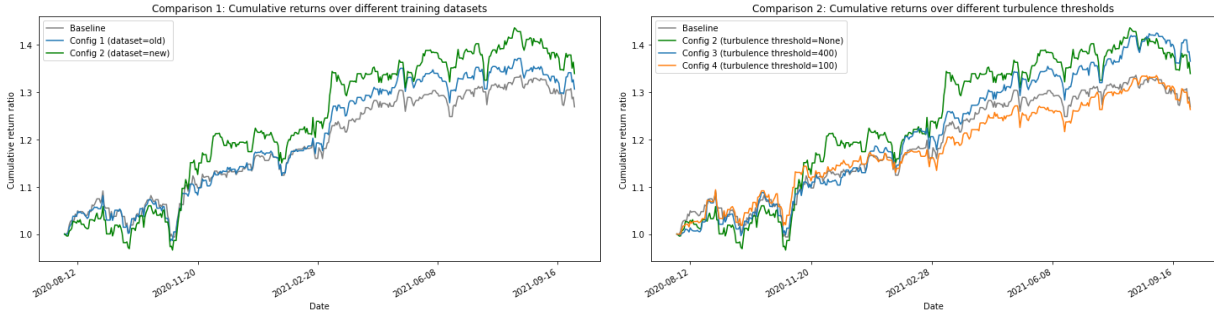
We design 6 sets of experiment configurations for agent training, each having different environmental attributes (shown in the table below). For each configuration, we train an agent on the training data and evaluate its performance on the testing data. We use the market index (DJIA) as our baseline. We choose Soft Actor-Critic (SAC) as the agent type for multiple stock trading because it is one of the most recent state-of-art algorithms and is characterized by its stability. In all evaluations, we start with \$1,000,000 initial capital at 2020-08-01. Then, we use each trained agent to trade the DJIA stocks and record a history of cumulative returns until 2021-09-30. The final return is the account value at 2021-09-30 divided by the initial capital. An additional distinction between the old dataset and the new dataset (shown in the table below) is that the old dataset only contains stock data while the new dataset is the one we described in our data contribution.

Config No.	Dataset version	Turbulence threshold	Transaction cost percentage	Final return
0	Baseline (DJIA market index)			1.269
1	old	None	0.001	1.307
2	new	None	0.001	1.340
3	new	400	0.001	1.366
4	new	100	0.001	1.264
5	new	None	0.01	1.292
6	new	None	0.02	1.299

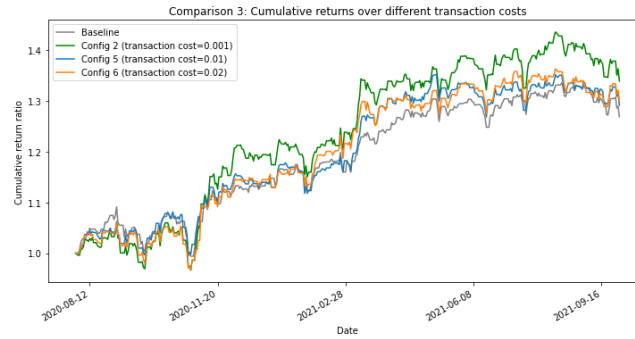
We set up 3 groups of comparisons using data collected from our experiments. Small figures are included in this section. See appendix for full-size figures.

**Comparison 1:** The evaluations of **Config No. 1 and 2** show that our new dataset successfully enables the agent to achieve a higher return.

**Comparison 2:** In evaluations of **Config No. 2, 3, and 4**, we observe that the agent with a turbulence threshold of 400, which is greater than 99% of the turbulence index of all stocks at any time, outperforms the agent without a threshold and the agent with a threshold of 100. The latter’s final return is less than but close to the baseline.



**Comparison 3:** In evaluations of **Config No. 2, 5, and 6**, we observe that higher transaction costs significantly reduce the overall return, yet still outperform the baseline. The transaction costs of 0.01 and 0.02 yield similar final returns.



## 6) Compute/Other Resources Used

We leverage the High-RAM premium GPU in Colab Pro. We quantify resources used in terms of Colab GPU compute units. All 3 teammates upgraded to Colab Pro, each providing 100 units. In addition, we purchased 100 units. In total, our experiments used up approximately 400 units.

## 7) Conclusions

We have the following conclusions according to our 3 groups of comparisons:

**Comparison 1** verifies that our new dataset with both stock market indicators and financial ratios successfully helps the agent perceive the stock environment, producing a higher return. It implies the benefits of supplementing stock trading with financial data analysis.

**Comparison 2** indicates that a reasonable turbulence tolerance, like 99% (threshold=400), is capable of demonstrating risk aversion (compared to no threshold which might suffer greatly from volatility) while not being too conservative. It is also worth noting that a turbulence tolerance of 90% yields a final return less than but close to the baseline, suggesting that using such tolerance level is potentially as rational as following the market. This analysis provides insights for investors.

**Comparison 3** indicates that at higher transaction costs, final returns shrink. We interpret this outcome from two perspectives. The commission fee takes most of the credits. In addition, the increase of cost forces the agent to operate more conservatively so that the agent misses some profitable opportunities that require a large volume of transactions.

Overall, our trained agents are able to beat the market index baseline in a more complex (similar to real) market. It suggests that automated stock trading has the potential to outperform traditional manual trading methods. The trained agents' capability to earn profits and the insights gained from simulations would enhance both individuals' and financial institutions' capabilities of investments and financial management. However, it is worth exploring the case where races occur when most of the market apply similar automated trading strategies. From an even more general view considering the entire global economy, we want to note that our trained agents will have limited power in reacting to potential recessions. If the market undergoes significant sudden decreases, the trained agents are expected to suffer from huge losses. As a future step, we could try supplying sentiments generated through web scraping to the training of agents and develop a reaction mechanism that helps avoid or minimize losses if signs of a recession are detected.

(Exempted from page limit) Other Prior Work / References (apart from Sec 3) that are cited in the text:

1. Leslie Kaelbling, Michael Littman, and Andrew Moore. "Reinforcement learning: A survey. " Journal of Artificial Intelligence Research, 4:237-285, 04 1996.
2. Yang, Hongyang and Liu, Xiao-Yang and Zhong, Shan and Walid, Anwar, Deep Reinforcement Learning for Automated Stock Trading: An Ensemble Strategy (September 11, 2020).
3. Xiao-Yang Liu, Hongyang Yang, Qian Chen, Runjia Zhang, Liuqing Yang, Bowen Xiao, Christina Dan Wang, FinRL: A Deep Reinforcement Learning Library for Automated Stock Trading in Quantitative Finance (September 2011).

(Exempted from page limit) **Broader Dissemination Information:**

Your report title and the list of team members will be published on the class website. Would you also like your pdf report to be published?

YES

If your answer to the above question is yes, are there any other links to github / youtube / blog post / project website that you would like to publish alongside the report? If so, list them here.




(Exempted from page limit) Attach your midway report here, as a series of screenshots from Gradescope, starting with a screenshot of your main evaluation tab, and then screenshots of each page, including pdf comments. This is similar to how you were required to attach screenshots of the proposal in your midway report.

## Project Milestone 2

● GRADED

GROUP

Chu'An Li  
Yicheng Shen  
Dingyuan Liu

 View or edit group

TOTAL POINTS

**7 / 7 pts**

QUESTION 1

**Project Milestone 2** 7 / 7 pts

✓ + 1 pt

Does the report follow the provided template including the 4-page limit (excluding exempted portions), with reasonable responses to all questions?

✓ + 2 pts

Has feedback from the last round been effectively addressed?

✓ + 1 pt

Has the team identified a clear topic and viable new target contribution, as per the project specifications provided in class?

✓ + 3 pts

Has the team moved in a non-trivial way towards their target contribution?

# Automating Stock Trading with Reinforcement Learning

**Team:** Chuan Li (CIS 5190), Dingyuan Liu (CIS 5190), Yicheng Shen (CIS 5190).

**Project Mentor TA:** Jason Ma

## 1) Introduction

**Set up the problem:** Our task is to maximize the cumulative return of a virtual portfolio containing a number of stocks. In this project, we focus on 30 stocks in the Dow Jones Industrial Average (DOW 30). To achieve this goal, we use reinforcement learning (RL) to train an automated trading agent on historical data which include both stock data and financial data from the past. We leverage the existing FinRL framework to train the agent [Xiao-Yang Liu, 2021]. In the ideal scenario, we expect the trained agent to perform profitable trading actions on new input data.

Essentially, the problem we aim to solve with RL can be described as a Markov Decision Process (MDP) [Leslie Kaelbling, 1996]. The optimization objective is to maximize the cumulative expected return. In our problem formulation, the main components are the following:

- **Environment:** The environment consists of stock and financial data of stocks in the DOW 30 along with hyperparameters that simulate a real market.
- **State space  $\mathcal{S}$ :** The state space represents all the information an agent can observe in the market environment. Each  $s \in \mathcal{S}$  represents a specific state in the environment. In our setup, every state  $s$  contains historical stock data and financial data on a specific date.
- **Action space  $\mathcal{A}$ :** The action space consists of all actions that an agent can perform. In our setup, each action  $a \in \mathcal{A}$  represents a set of trading decisions which includes selling, holding, and buying a number of shares of a stock or multiple stocks.
  - Example: Suppose we only consider one stock AAPL. We hold 100 shares of it and there are  $n$  shares on the market. In this case, the action space  $\mathcal{A}$  is  $\{-100, \dots, -1, 0, 1, \dots, n\}$ . An action  $a \in \mathcal{A}$  could be selling at most all 100 shares ( $a = i, i \geq -100$ ), holding the current shares ( $a = 0$ ), or buying a number of shares ( $a = i, i \leq n$ ).
- **Reward function  $\mathcal{R}$ :** The reward function  $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$  maps a tuple  $(s, a)$  consisted of a state and an action to a reward  $r$  which is a real value. It incentivizes the agent to learn a policy that generates a higher return. In our setup, the reward at time  $t$  is the difference between the return in state  $s_t$  and the return in state  $s_{t+1}$  after the action  $a_t$  is taken.
- At time  $t$ , the objective that we want to maximize can be written as

$$E\left(\sum_{t=0}^{\infty} \gamma^t r_t\right)$$

where  $r_t = \mathcal{R}(s_t, a_t)$  and  $\gamma$  is the discount factor ( $0 \leq \gamma \leq 1$ ).

Based on the problem formulation stated above, we can train our automated trading agent using historical stock data and financial data. The training set includes data from 2009 to 2018. The backtesting set contains data from 2019 to 2021. For the trained agent, we input new stock and financial data, and the agent outputs its decisions based on the policy it learned. To evaluate the agent's performance, we will use the cumulative return and the daily return compared with the market index as our major performance metrics.

**Motivation:** If we are able to train an automated stock trading agent using RL, the steps we take do not only apply to the DOW 30 stocks strictly. Potentially, this could scale up and perform decision making in the real market. Our project could also show an analysis of different choices of RL algorithms to provide investors with some insights on the performance of different algorithms. If this project succeeds in maximizing the value of any virtual portfolio with a performance better than regular methods, it would enhance the investment and financial management capability of both individuals and organizations.

## 2) How We Have Addressed Feedback From the Proposal Evaluations

The key feedback from our proposal is that the scope of our project defined in the proposal is too broad to accomplish given the limited time. The feedback suggests that we should leverage existing stock market simulators to train a good RL policy and carefully document our approaches. According to the great suggestions in the feedback, we revised our scope and formed a clear goal to focus on FinRL to train our trading agent. The existing FinRL framework will help us alleviate implementation and debugging workload, thus we could focus on developing the RL policy for maximizing returns. We will also thoroughly document the process of training the policy and conduct careful evaluations on our result.

## 3) Prior Work We are Closely Building From

- A. Xiao-Yang Liu and Hongyang Yang and Jiechao Gao et al, "FinRL: Deep Reinforcement Learning Framework to Automate Trading in Quantitative Finance", ICAIF 2021. Github implementation at: <https://github.com/AI4Finance-Foundation/FinRL>.  
FinRL is a deep reinforcement learning (DRL) library that focuses on stock trading. It consists of three layers: i) an environment layer that simulates the financial market constructed with historic data, ii) an agent layer that provides DRL algorithms, and iii) an application layer for automated stock trading. This framework serves as the backbone in our project to train an automated stock trading agent.
- B. Stock\_Fundamental.ipynb. [https://github.com/AI4Finance-Foundation/FinRL/blob/master/tutorials/1-Introduction/Stock\\_Fundamental.ipynb](https://github.com/AI4Finance-Foundation/FinRL/blob/master/tutorials/1-Introduction/Stock_Fundamental.ipynb).  
This notebook contains a basic pipeline of using FinRL to train a trading agent. It helps us get started with using the FinRL framework and develop our own trading policy.

## 4) What We are Contributing

- 
1. **Contribution(s) in Application/Data:** We add new data for training our trading agent. We collected historical financial data from Wharton Research Data Services (WRDS). We use the new data along with historical stock data to set up the market environment.
  2. **Contribution(s) in Algorithm:** We aim to improve performance by transforming the data on financial statements into useful ratios and use them as features in training. We will evaluate whether the additional features will help our trading agent's decision making.
  3. **Contribution(s) in Analysis:** We analyze the agent's performance of cumulative return in environments with different attributes such as transaction cost and turbulence threshold.

#### 5) Detailed Description of Each Proposed Contribution, Progress Towards It, and Any Difficulties Encountered So Far

##### 5.0.1 Detailed Description of Contributions in Data:

We leverage Wharton Research Data Services (WRDS) and search for financial data to complement the stock data from Yahoo Finance. WRDS is a business data research service from The Wharton School and it provides data access for academic research purposes. We find the Financial Ratios Firm Level database to be a good fit for our project. Since we focus on stocks in the DOW 30, we query the database by their stock ticker in the date range of 2009-01-01 to 2022-10-31. We obtain monthly financial data (containing null values) that indicate companies' valuation, profitability, liquidity, etc.

The new data contains 4817 rows and 78 columns. Among the columns, 4 are identifiers, 3 are dates, and 71 are financial ratios, such as *net profit margin*, *return on equity*, etc. A number of columns contain roughly 15% of null values. The column with the most null values contains more than 30% of null values. The challenge is to figure out the right condition to drop a column. Note that since we are dealing with stock and financial data which are time series, we cannot simply drop rows with null values. Currently, we set the tolerance for null values to be 1% and we have 40 columns of financial ratios left. For the very few null values left, we fill them with 0.

Another decision we made with the data is to average monthly data into quarterly data. This manipulation of the data aligns with the real world timing because companies' financial statements are usually released quarterly. This also helps to solve the issue of null values because some null values could be filled in by the average values.

After all the preprocessing, we merge the new financial data with the stock data and create a DataFrame that is ready for EDA, feature engineering, and setting up the RL environment.

##### 5.0.2 Detailed Description of Contributions in Feature Engineering:

Financial ratios are created with the use of numerical values taken from financial statements to gain meaningful information about a company. Determining individual DOW 30 stocks financial ratios per period and tracking the change in their values over time is done to spot trends that may be developing in a company. Comparing financial ratios with that of major competitors is done to identify whether a company is performing better or worse than the industry average.

---

While 71 financial ratios were included, 5 ratio categories can summarize them all. Those 5 categories are Liquidity ratios, Leverage ratios, Efficiency ratios, Profitability ratios, as well as Market value ratios. For each category, there are some common and important ratios. Take Liquidity ratios as example: it contains current ratio, cash ratio, operating cash flow ratio etc. Some ratios stem from others, resulting in correlations among some ratios. Thus, involving more ratios and making sure every involved ratio matters would be essential for feature engineering. The first step of feature engineering is to select one centroid of the five clusters using finance interpretation. For instance, debt ratio would always be included in Leverage ratios. Then set the rest of the ratios as features of PCA. To determine an appropriate  $n\_components$ , plot cumulative sum of explained variance ratios and also a line for 0.95 variance, a general benchmark (plot attached at the supplement information section). When  $n$  components were around 20, the explained variance ratios started to plateau. With the right amount of components capturing the majority of variance, we mimic the market environment more rigorously.

#### 5.0.3 Detailed Description of Contributions in RL Performance Analysis on Environmental Attributes:

Previous work set up a single environment containing fixed turbulence threshold which controls agent risk aversion and consistent transaction cost which limits the total amount of agent's operations. This lack of environmental variety conceals the potential of RL agents. We expect RL agents to perform better than baseline when the environment is less ideal and more complicated.

We supplement previous work analysis by tuning turbulence thresholds to simulate environments of different degrees of volatility. We also adjust the transaction cost according to the real-world situation. We plan to compare the performance under a turbulence threshold space of 6 and a transaction cost space of 6.

A turbulence threshold in a given environment is a user-defined scalar which controls the agent's buy and sell actions. In any given state, the agent extracts turbulence indexes from every stock's data and compares them with the turbulence threshold to determine which of the stocks are volatile. If true, the agent would avert risky operations. For example, the agent would not buy any shares of a certain stock if it senses it as volatile regardless of current price.

A transaction cost is also a user-defined scalar representing the transaction fee of both buy and sell operation. It is deducted from the reward function in every non-hold action. The change in transaction cost may influence the agent's strategy in terms of stock selection and operation period.

#### 5.1 Methods

At the end of day of given time  $t$ , we will know the open-high-low-close price of the Dow 30 constituents stocks, and we refer those data as state  $s$ . Then, we feed the states into our FinRL



model, the trader will output a list of actions, treating this value as the trading signal to buy or sell [Hongyang Yang, 2011]. Actions to trade would be based on DRL suggestions, expecting the cumulative shares would grow at the end of day. We then add the shares to buy to shares previously held, while subtract shares to sell to get updated shares held. Then at time  $t + 1$ , following the same process, we can calculate the reward. The reward at time  $t$  is the difference between the return in state  $s_t$  and the return in state  $s_{t+1}$  after the action  $a_t$  is taken. The reward can be positive or negative, and we want to train DRL to only learn from positive rewards to be effective. The whole procedure would repeat until termination criteria reached.

## 5.2 Experiments and Results

The key question of our experiment is how amplified financial data (contributions in data collection and feature engineering) and empirical environmental attributes (contributions in performance analysis) might improve the RL agent performance in terms of cumulative return over a certain period. We expect the RL agent to perform better when it is exposed to more information provided by WRDS than mere Yahoo Finance. We also anticipate there exists an approximated combination of turbulence threshold and transaction cost that best describe the real world environment hence boost RL agent's understanding of the stock trading market resulting in a greater cumulative return.

We compare our approach to the baseline provided by the FinRL backtesting module. Backtesting plays a key role in evaluating the performance of a trading strategy. Automated backtesting tools are preferred because they reduce human error. We use the Quantopian pyfolio package to backtest trading strategies. It is easy to use and consists of various individual plots that provide a comprehensive image of the performance of a trading strategy.

### 6) Risk Mitigation Plan

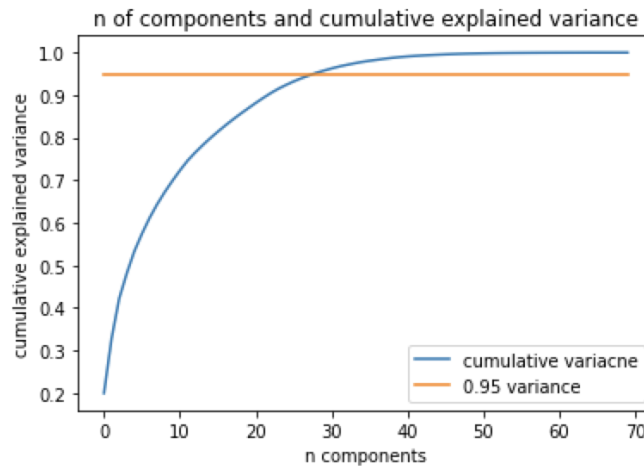
Since we have contributions on both the data part and analysis part, we could still produce a viable report if one of them fails. The chance of an overall failure is extremely low. Additionally, if amplified datasets do not significantly improve the RL agent's performance, we could still do feature prune or selection to explore beneficial subsets against noisy ones. For the analysis part, we could always discuss and reason about the outcome because the RL performance over 2-dimensional parameter space usually demonstrates some specific patterns.

(Exempted from page limit) Other Prior Work / References (apart from Sec 3) that are cited in the text:

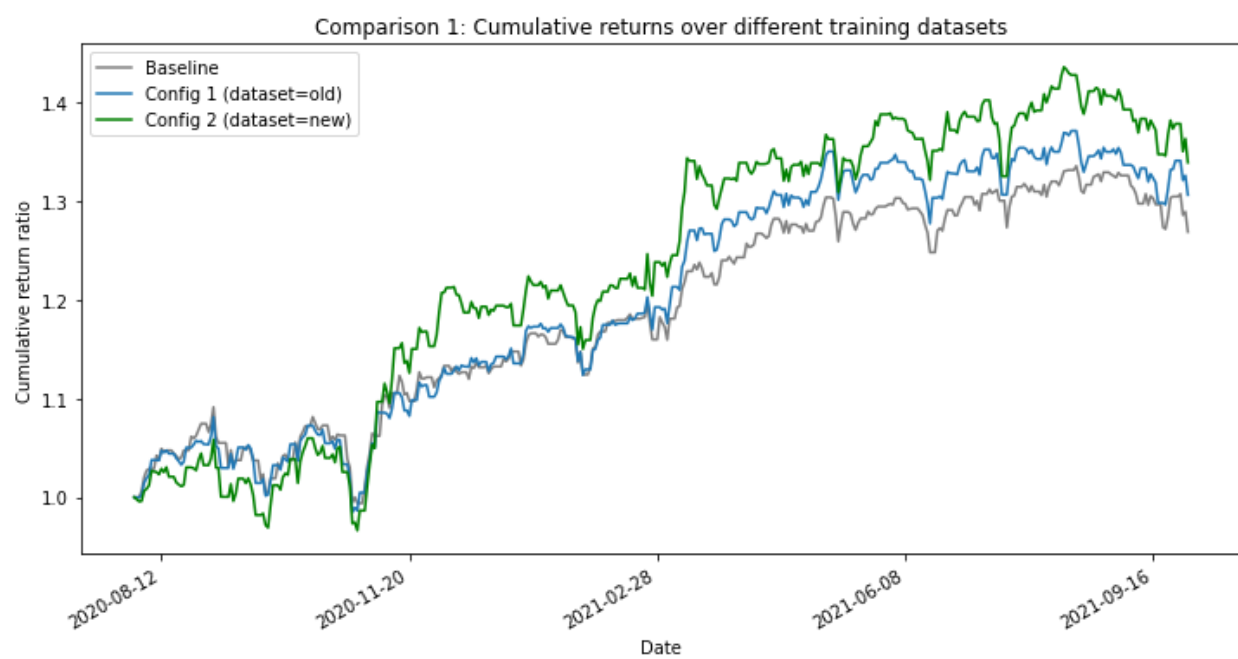
1. Leslie Kaelbling, Michael Littman, and Andrew Moore. "Reinforcement learning: A survey." *Journal of Artificial Intelligence Research*, 4:237-285, 04 1996.
2. Yang, Hongyang and Liu, Xiao-Yang and Zhong, Shan and Walid, Anwar, Deep Reinforcement Learning for Automated Stock Trading: An Ensemble Strategy (September 11, 2020).
3. Xiao-Yang Liu, Hongyang Yang, Qian Chen, Runjia Zhang, Liuqing Yang, Bowen Xiao, Christina Dan Wang, FinRL: A Deep Reinforcement Learning Library for Automated Stock Trading in Quantitative Finance(September 2011)

(Exempted from page limit) Supplementary Materials if any (but not guaranteed to be considered during evaluation):

Plot for PCA mentioned in 5.0.2 Detailed Description of Contributions in Feature Engineering:

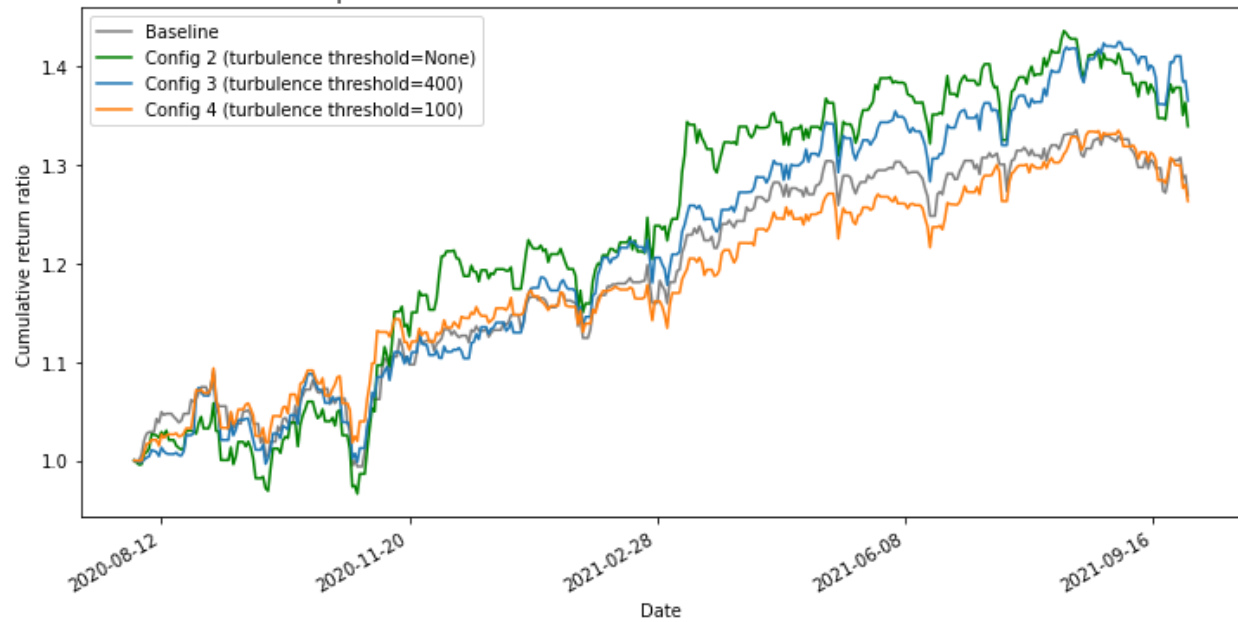


## Full-size figures for 5.2 Experiments and Results:





Comparison 2: Cumulative returns over different turbulence thresholds



Comparison 3: Cumulative returns over different transaction costs

