Team 8 Project:
Deep Reinforcement Learning for Automated Stock Trading in Quantitative Finance

Muhao Guo
1221658724

Jiaqi Wu
1217451339

Zhihao Ma
1209635632

***Abstract*—Automated stock trading is a popular topic of investment. Recent work has shown the success of machine learning techniques in predicting stock prices. This work explores different automated trading algorithms based on Deep Reinforcement Learning (DRL) models. Five independent Actor-Critic (A2C/DDPG/PPO/TD3/SAC) algorithms and one ensemble learning method are considered, with maximum portfolio value as the target of Reinforcement Learning (RL) reward function. To find out the indicator's performance, we compare the results of the algorithms with and without indicators. Experiment results show that indicator is effective to predict. These DRL-based algorithms show their effectiveness by outperforming Dow Jones Industrial Average index, evaluated by the Sharpe ratio.**

1. Introduction

The automated trading system plays a growing essential role in stock trading with faster reaction and time-efficiency operations in the modern trading market. Common stock operations can be summarized as holds, sells, buys, and transformed into a trading model for automated trading systems. Compared with traditional models, such as Modern Portfolio Theory (MPT) and machine learning methods for predicting stocks prices, Reinforcement learning (RL) based methods are milestones of the automated trading system. RL-based algorithms can make the most of large-scale data and be robust to outliers and automatically design different strategies based on the trained targets. However, as humans do not thoroughly research RL, the RL-based automated trading system is still challenging for the actual application. In this project, we explored and analyzed RL-based algorithms for decision-making in stock trading analysis.

Common stock operations can be summarized as hold, selling, and buying, which can be converted into a trading model for an automated trading system. We use different practical algorithms to predict the stock prices and decide based on Deep Reinforcement Learning (DRL). RL-based algorithms can make the most of large-scale data and be robust to outliers and automatically design different strategies based on the trained targets. In order to archive these procedures, we utilize the FinRL library [1] to customize our state space, action space, reward function, which will be described in our latter sections. Our dataset is Dow Jones 30 stock's price data from 2009 to the present. To find out which algorithm has a better performance in this problem, we compare the final cumulative return of different algorithms with the Dow Jones Industrial Average (DJIA) and evaluate each algorithm using the Sharp ratio [2] with and without the indicators.

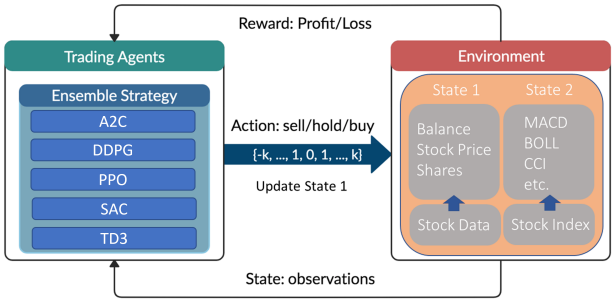

Figure 1. The architecture of DRL-based stocking trading algorithm.

2. Methods

2.1. Environment Setup

Since we use Dow Jones 30 stocks, the definitions of the environment and the corresponding data dimension are shown in the following subsections.

2.1.1. State Space $\mathcal{S}$. State Space contains a set of states which are 61-dimension, 181-dimension or 241-dimension vectors which can be described as $\{b_t, \boldsymbol{p}_t, \boldsymbol{h}_t, \mathbf{I}_t\}$ where

- Balance $b_t \in \mathbb{R}_+$ is the available balance at time step t.
- Closing price $\boldsymbol{p}_t \in \mathbb{Z}_+^{30}$ is the vector of close price of each stock.
- Shares own $\boldsymbol{h}_t \in \mathbb{R}_+^{30}$ is the vector of shares owned of each stock.
- Indicator $\mathbf{I}_t$ is the vector of indicators of each stock. We plan to use a indicator vector which includes indices of Moving Average Convergence Divergence (MACD), Bollinger Bands Index (BOLL), Commodity Channel Index (CCI), Relative Strength Index(RSI), Average Directional Movement Index(ADX), etc. Specifically, $\mathbf{I}_t = \{\mathbf{I}_{1t}, \mathbf{I}_{2t}, \ldots, \mathbf{I}_{nt}\}, \mathbf{I}_{1t}, \mathbf{I}_{2t}, \ldots, \mathbf{I}_{nt} \in \mathbb{R}_+^{30}$. Those technical indicators will be calculated by history open-high-low-close price of the stocks.

2.1.2. Action Space $\mathcal{A}$. For each stock, the action is $a = \{-k, \ldots, -1, 0, 1, \ldots, k\}$, where positive action means buying, negative action means selling and 0 means holding. Additionally, k denotes the number of shares. In practise, action space will be normalized between 1 and -1.

2.1.3. Reward Definition $r(s, a, s')$. Reward is the single-step reward value for action a is taken at state s and arriving at new state s'. In this project, reward $r(s, a, s')$ is defined as $r(s, a, s') = v' - v$, where v' and v represent the portfolio values at state s' and s, respectively.

2.1.4. Assumptions. In order to simplify the calculation, we made the following assumption.

- Only one action in one day.
- All stocks trading are taken in one action.
- Trade prices are the same as the close price in the same day.
- Action space was normalized between -1 and 1 as the policy distributions are defined as a Gaussian distribution.

2.1.5. Evaluation. To evaluate the performance of each algorithm, We will not only compare the final balance results of different algorithms with the DJIA, but also evaluate each algorithm using the Sharp ratio. The Sharpe Ratio is a standardized indicator used in quantitative finance for considering the balance between rewards and risks. The Sharpe Ratio has been studied in modern investment theory, showing that the magnitude of risk plays a fundamental role in determining the performance of a portfolio. The sharp ratio can be algebraically described as

$$Sharpratio = \frac{\bar{r}_p - r_f}{\sigma_p} \tag{1}$$

where $\bar{r}_p$ is the expected portfolio return, r_f is the risk free rate, and σ_p is the portfolio standard deviation.

2.2. Reinforcement Learning Algorithms

The key references of our project are [1] and [3]. [1] is to illustrate the formulation of applying DRL method for automated stock trading and [3] is to introduce the library which is FinRL, used in this project. We will mainly implement five different Deep Reinforcement Learning (DRL) algorithms for stock trading agents, which are Advantage Actor-Critic (A2C) [4], Deep Deterministic Policy Gradient (DDPG) [5], Proximal Policy Optimization (PPO) [6], Soft Actor-Critic (SAC) [7] and Twin Delayed DDPG (TD3) [8]. All of them are Actor-Critic RL algorithms.

One Actor-Critic algorithm, shown in Fig. 2, is a combination of policy-based and value-based algorithms. It has two parts, the Actor and the Critic. The Actor uses a policy function responsible for taking actions and interacting with the environment. The Critic uses the value function, which is responsible for evaluating the Actor's performance and guiding the Actor's actions in the next stage [9].

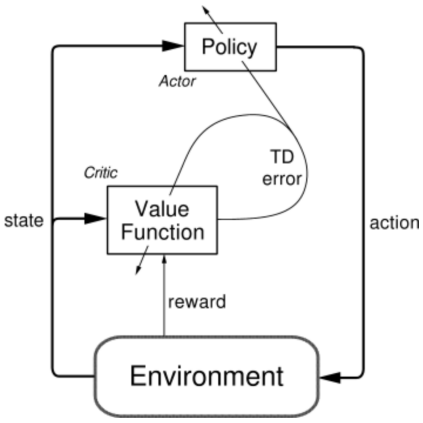

Figure 2. The actor-critic architecture.

2.2.1. Advantage Actor-Critic (A2C). For A2C, multiple workers in parallel environments independently update a global value function with the synchronized update. A2C updates the actor network by awarding good actions, using the MC method minus baseline to estimate Q-function.

2.2.2. Proximal Policy Optimization (PPO). PPO is a family of methods that approximately enforce KL constraints without computing natural gradients. It has two variants, which are Adaptive KL Penalty and Clipped Objective. In this project, Clipped Objective is chosen as our consideration.

2.2.3. Deep Deterministic Policy Gradient (DDPG). DDPG is an algorithm closely connected to Q-learning and concurrently learns an approximator to $Q^*(s, a)$ with an approximator to $a^*(s)$. It uses off-policy data and the Bellman equation to learn the Q-function, and the Q-function to learn the policy.

2.2.4. Soft Actor-Critic (SAC). Soft Actor-Critic (SAC) is based on a maximum entropy RL where the objective is to find the optimal policy that maximizes the expected long-term reward and long-term entropy. It does it by making use of three networks: a state value function V parameterized by Φ, a soft Q-function Q parameterized by θ, and a policy function π parameterized by Φ and minimizing the corresponding errors.

2.2.5. Twin Delayed DDPG (TD3). TD3 is based on DDPG and addresses the problem of continuously overestimating the Q values of the critic network in DDPG by reducing the overestimation bias. It mainly adds three key features: two critic networks inspired by double Q-learning, delayed actor updates, and action noise regularization target policy smoothing.

2.2.6. Ensembled Algorithm. Further, an ensemble algorithm is implemented to make trading more reliable and risk-aware. The ensemble-learning algorithm inherits and

integrates the best performance of each sub-DRL algorithms and slides across all the trading time to dynamically adjust different trade situations. The index to evaluate the algorithm is the Sharpe ratio, the average return earned more than the risk-free rate per unit of volatility or absolute risk.

3. Implementation and Simulation

3.1. Introduction of FinRL

FinRL [1], [10] is a open-source Deep Reinforcement Learning Library for automated stock trading. This FinRL framework consists of three layers: environment, agents, and applications. The environment layer includes build-in data collection APIs and live trading APIs, reconfiguring into a standard OpenAI gym-style environment. The agents layer provides a variant of DRL algorithms from ElegantRL [11], Ray RLlib [12] and Stable Baselines 3 [13], which we can easily change the hyperparameters. The application layers allow users to customize FinRL for different stock trading tasks.

3.2. Dataset

As shown in Fig. 3, our dataset is Dow Jones 30 stock's price data from 2009 to the present, and we divide it into three separate sections for our training, validation, and testing. Data from 2009 to 2015 is selected for training, while data from 2015 to 2016 is selected for validation and data from 2016 to present is utilized for testing. The dataset is obtained from Yahoo Finance API provided by FinRL and can also be downloaded from Kaggle [14].

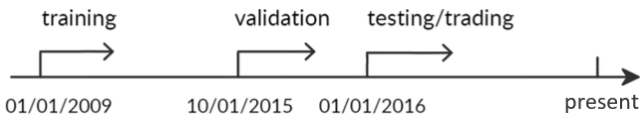

Figure 3. The time frame selection for our training, testing and validation dataset.

3.3. Installation

To start with, we run the code in Google Colab. To speed up, here, we can use the GPU. Then we cloned the repository and installed the unstable development version of FinRL [10] using pip.

3.4. Prerequisites and Dependencies

Since the Colab is based on Ubuntu, we run the code on a Ubuntu system. To avoid packages conflicts, we created a Python Virtual Environment using pip; the result shows in Fig. 6. We also installed some system packages like CMake, OpenMPI, and Zlib; see Fig. 5. Before running the code, there are several dependencies shown in Fig. 7 which need to be installed using pip under Python $\geq$ 3.6.0.

Figure 4. FinRL was successfully installed.

Figure 5. Installed the system package successfully.

3.5. Training Results

First, we need to loading the datasets. Fig. 8 shows we successfully fetch our data and do feature engineering. Then, we run the code, Fig. 9 shows the training and the backtest process.

4. Results

4.1. Four indicators will not be helpful

To get the cumulative returns, we use DDPG with four indicators, DDPG without indicators, A2C with four indicators, and A2C without indicators, respectively. The results in Fig. 10 show that every algorithm except the A2C with four indicators can achieve more cumulative returns than the actual returns.

Then, we compared five algorithms, including the DDPG with four indicators, DDPG without indicators, A2C with four indicators, A2C without indicators, and the ensemble learning-based algorithm. Fig. 11 shows the comparison. From the result, we get that the A2C without indicators is the best algorithm among the four algorithms we use. The DDPG with four indicators and DDPG without indicators have similar cumulative returns in the end, which illustrates that the DDPG algorithm is not sensitive to the indicators.

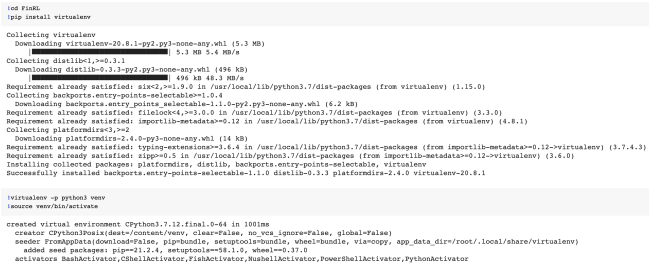

Figure 6. The virtual environment was successfully created and activated.

```
numpy>=1.17.3
pandas>=1.1.5
stockstats
yfinance
#pyfolio
matplotlib
scikit-learn>=0.21.0
gym>=0.17
stable-baselines3[extra]
ray[default]
ray[tune]
lz4
tensorboardX
gputil
trading_calendars
alpaca_trade_api
ccxt
jqdatasdk
wrds
pytest
setuptools>=41.4.0
wheel>=0.33.6
```

Figure 7. requirements

The A2C with four indicators is the worst among the four algorithms, demonstrating that the A2C algorithm is sensitive to the indicators. When adding the four indicators, even though the complexity of the model has increased, it is helpless even has a bad effect on the effect of the model. The ensemble-learning-based algorithm can hold a balance in the different trends of marketing.

4.2. Six indicators are partially helpful

When we increase the number of indicators to six, some algorithms will have potential advantages. In Fig. 13, we show the five algorithms, including the A2C, PPO, DDPG, SAC, and TD3, all of them are with six indicators. In Fig. 14, all the five algorithms are without indicators. In Fig. 12, we compare all the ten algorithms in Fig. 13 and Fig. 14 with the Dow Jones. We know from the numerical results that all the ten algorithms can get more cumulative returns at the end of the date. However, the six indicators will only be helpful with A2C and PPO algorithms. For the DDPG, SAC, and TD3 algorithms, adding the six indicators as data features will not be helpful to improve the performance of the original algorithms.

Finally, we use the ensemble algorithm, which combines the five algorithms in the data training process and chooses the best one as the target algorithm in any time slot. The result in Fig. 15 shows that the ensemble algorithm can also achieve a good performance in cumulative returns.

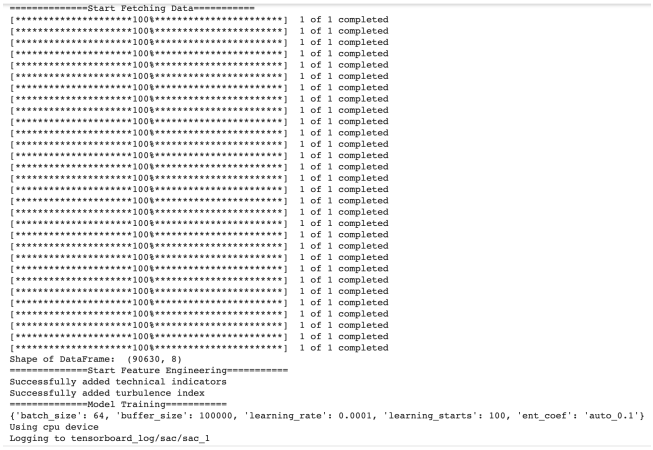

Figure 8. Successfully download the dataset and load into the model for training.

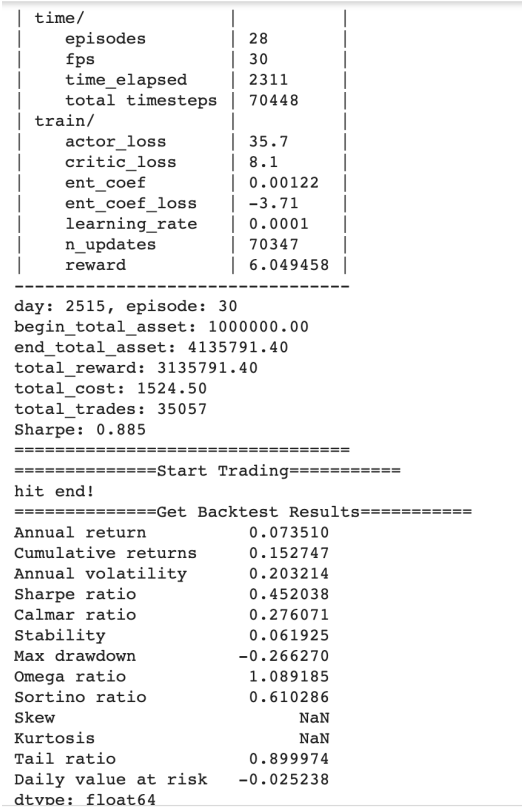

Figure 9. Successfully run the code for model training. This picture shows the results after 30 episode. The bottom half of the picture shows the backtest results.

5. Discussion and Conclusion

This project explores using Deep Reinforcement Learning algorithms to realize automated stocking trading. Table 1 shows detailed experience during this project.

By applying multi-methods with/without indicators, the experiments show that off-policy algorithms perform better than on-policy algorithms. However, adding entropy loss can

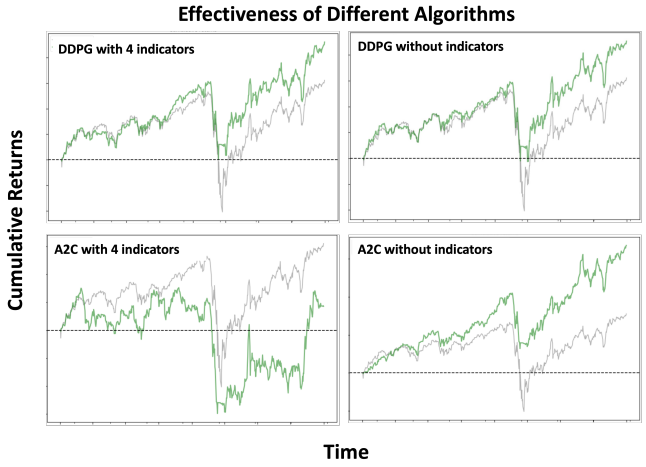

Figure 10. The four algorithms that we use, which are DDPG with four indicators, DDPG without indicators, A2C with four indicators, and A2C without indicators, respectively. The green line represents the cumulative return that our algorithm gets. The gray line represents the real cumulative return.

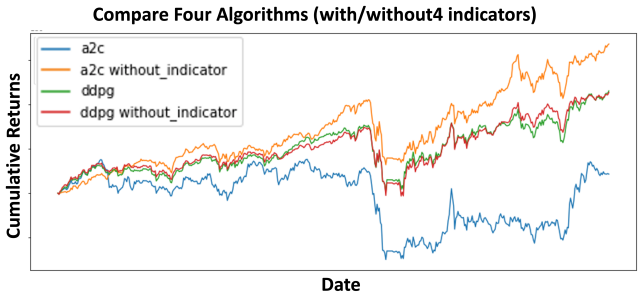

Figure 11. The comparison of the four algorithms. For any algorithm, without indicators means we do not use any indicators as a feature. Otherwise means we use four indicators.

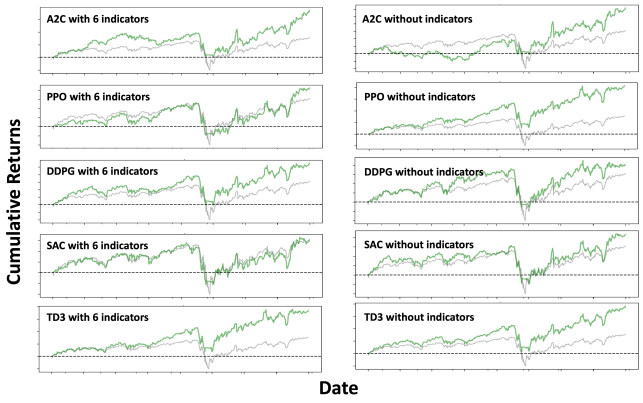

Figure 12. The cumulative returns for ten algorithms compared with the Dow Jones, including five algorithms with six indicators and five algorithms without indicators.

distinctly increase the final performance.

Simulation results show that the four indicator-assisted

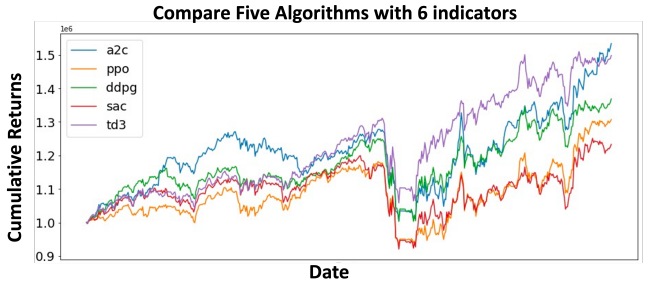

Figure 13. The comparison of the five algorithms. For any algorithm, six indicators are considered as data features.

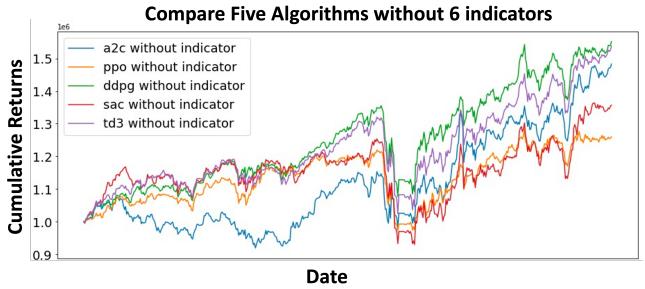

Figure 14. The comparison of the five algorithms. For any algorithm, the indicators are considered as data features.

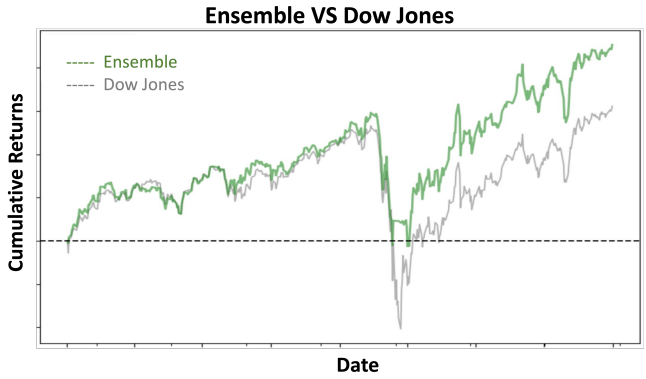

Figure 15. The cumulative returns for the ensemble algorithm compared with the Dow Jones.

methods will not own any advantages over algorithms without indicators. Moreover, we try more indicators, such as six indicators as the features. We believe as the number of indicators increases. It can improve the performance of the algorithms because the indicators can be seen as some essential features. However, the performance with six indicators does not significantly impact the result as we expected. Moreover, only two algorithms have benefited from having six indicators, while the remaining three algorithms have worse performance with six indicators than without them.

Further, we find ensemble learning-based method can balance different marketing trends, whether in a bullish trend or a bearish trend. However, the current ensemble design is

not robust enough. We attribute to the insufficient data of sliding window. We would explore an incremental window to realize a better ensemble learning algorithm in the future.

6. Individual Statement of Contributions

Each of us is in charge of one major part (methodology, data, and code), and we discussed a lot to ensure team corporation on all the tasks. At each group meeting, we check the progress and make sure all the required parts are finished.

Especially, Zhihao, Jiaqi, and Muhao equally shared the workload in the final presentation, preparing slides and rehearsing several times. Moreover, we worked together on the report construction. While all the group members have reasonable control over the contents, Zhihao focuses on the introduction and the logic transitions among sections, Jiaqi focuses on the demonstration of the method, Muhao focuses on experiment setups and deliverable results.

- **Abstract & Introduction:** Zhihao leaded this part and completed 70%, Jiaqi completed 10% and Muhao completed 20%.
- **Method**: Jiaqi leaded this part and completed 75%, Muhao completed 10%, and Zhihao completed 15%.
- **Implementation and Simulation & Results**: Muhao leaded this part and completed 60%, Jiaqi completed 25%, and Zhihao completed 15%.
- **Discussions and Conclusions:** Zhihao completed 33%, Jiaqi completed 33% and Muhao completed 33%.

References

[1] X.-Y. Liu, H. Yang, Q. Chen, R. Zhang, L. Yang, B. Xiao, and C. D. Wang, "FinRL: A deep reinforcement learning library for automated stock trading in quantitative finance," *arXiv preprint arXiv:2011.09607*, 2020.

[2] W. F. Sharpe, "The sharpe ratio," *Journal of portfolio management*, vol. 21, no. 1, pp. 49–58, 1994.

[3] H. Yang, X.-Y. Liu, S. Zhong, and A. Walid, "Deep reinforcement learning for automated stock trading: An ensemble strategy," *Available at SSRN*, 2020.

[4] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[5] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[6] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[7] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[8] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1587–1596.

[9] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Advances in neural information processing systems*, 2000, pp. 1008–1014.

[10] "FinRL: Deep reinforcement learning for quantitative finance." [Online]. Available: https://github.com/AI4Finance-Foundation/FinRL

[11] "ElegantRL: Scalable and elastic deep reinforcement learning." [Online]. Available: https://github.com/AI4Finance-Foundation/ElegantRL

[12] "Ray RLlib." [Online]. Available: https://docs.ray.io/en/master/rllib/index.html

[13] "Stable baselines 3." [Online]. Available: https://github.com/DLR-RM/stable-baselines3

[14] "Reinforcement learning stock trading." [Online]. Available: https://www.kaggle.com/alincijov/trading

Topic	Specific & detailed evidence is required to support claims
Literature (not well written or self-contained, not specific on implementation, no data source indicated, no source code indicated. . .)	The key reference paper uses lots of actor-critic algorithms that we did not learn during the class. Therefore, we spent a lot of time reading related papers and blogs to understand actor-critic algorithms. Besides, we spent a lot of time understanding the original example codes to modify these codes.
Setting up the environment and obtaining data	To make the whole project run successfully and avoid packages conflicts, we created a virtual environment using "virtualenv" (See Fig 6). There are several ways we can use to obtain the data. We get the data directly from FinRL package since this way is more convenient. (See Section 3.1)
To have the first successful test run (issues while debugging, compatibility problems. . .)	We meet several compatibility problems. The latest packages are different from the code where the paper claimed. Because the authors upgraded the code several times and published a new version recently. So the most important thing we should do is to search for the correct API and built-in functions in the latest version of FinRL. Then we apply the algorithms and train the data. In the end, we designed our way of presenting the results (See Fig.10 and Fig.11).
Obtaining results (algorithm/method is difficult to implement, hyperparameters difficult to tune...)	During the simulation, we found the original example codes can not run successfully. Thus, we have to edit them by ourselves to obtain a better result, including implementing different methods, plotting better result figures, and adjusting the dimension of input vectors.
Obtaining results (cannot duplicate what was reported in the paper, if so, why?)	We duplicate the result in the paper with a new finding. In the paper, the author mentions that the indicators for the stock market are useful for RL training. However, we found out that is not the case. Only two indicators (a2c & PPO) positively affect the RL training.
Reporting (Intro, Method, Result, Discussions, . . .)	With a good layout of the report is crucial and extremely important. The introduction to the discussion has a really clear section by section, which is highly beneficial for our final presentation. Readers can easily read and understand the reinforcement learning that we used in our experiment.

TABLE 1. LESSONS LEARNED FROM WORKING ON THE FINAL PROJECT.