An Automatic Portfolio Management
A Reinforcement Learning Approach

Liubov Tovbin

Computer Engineering Department
San Jose State University, San Jose, CA
Email: liubov.tovbin@sjsu.edu

Table of Contents

***Abstract*—When it comes to investment, an asset allocation strategy is crucial for success. In this project, the author studies the reinforcement learning approach for automated portfolio management. An ensemble of Advantage Actor-Critic, Proximity Policy Optimization, and Deep Deterministic Policy Gradient is a baseline method for this work. The author challenges the baseline method performance by introducing various modifications to it. The results show that the original method remains the optimal one, judging by the cumulative portfolio value over time.**

***Index Terms*—portfolio management, asset allocation, automatic stock trading, reinforcement learning, PPO, A2C, DDPG**

I. Introduction

Nowadays, automatic stock trading is a common practice. Financial security is a basic human need. The right trading strategy is financially rewarding, whereas a bad one could be devastating for an investor. Thus, it is no surprise that investors have always tried to leverage technology when making financial decisions.

The researchers approach finding the best investment strategy, along with stock prices prediction, quite often. The stocks' historical data only makes sense when viewed as a time series. Thus, to predict future stock prices, the researchers often chose algorithms designed to handle sequenced data such as Auto-Regressive Moving Average, Hidden Markov Model, and Recurrent Neural Networks. [1].

Reinforcement Learning (RL) is another approach that is suitable for machine learning from sequenced data. However, instead of predicting the stock price, we aim to discover the best action strategy to maximize a reward. An ability to apply the optimal trading strategy provides an obvious financial benefit for an investor.

In portfolio management, our goal is to preserve the initial investment value and continuously increase capital gains. In this project, the author studies the state-of-the-art approaches for an automatic portfolio management. The portfolio here is a collection of stocks that constitute the Dow Jones Industrial Average index with an initial value of one million dollars. The baseline model is adopted from (Yang et al., 2020) [2].

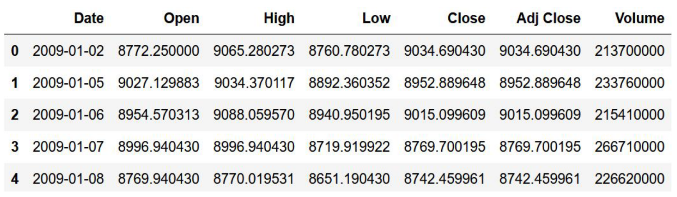
In this work, the author applies several modifications to the baseline algorithm that challenge it from various angles: independent variables, evaluation method, trading strategy under specific market conditions, and RL algorithm choice. The rest of the paper organized as follows:

- Section II gives an overview of the RL methods for an automatic stock trading in the literature.
- Section III thoroughly describes the baseline approach.
- Section IV explains the proposed modifications to the baseline approach and the reason behind them.
- Section V discusses the results.
- Section VI concludes this work and gives ideas for future research.

II. RL Methods for Stock Trading: Literature Review

To build an automatic stock trading system using the RL methods, we have to formulate the RL problem properly. According to the RL formalism, we need to define the environment representation, the action space, the reward, and the observations. In our case, we assume a fully observed environment, meaning the observation would be an environment state vector.

The choice of the above RL formalism entities depends on the data. Fig. 1 depicts a snapshot of a typical daily stock prices records. The data is a several continuous time series corresponding to the open, close, daily high and low stock prices along with the adjusted close price and the volume.



	Date	Open	High	Low	Close	Adj Close	Volume
0	2009-01-02	8772.250000	9065.280273	8760.780273	9034.690430	9034.690430	213700000
1	2009-01-05	9027.129883	9034.370117	8892.360352	8952.889648	8952.889648	233760000
2	2009-01-06	8954.570313	9088.059570	8940.950195	9015.099609	9015.099609	215410000
3	2009-01-07	8996.940430	8996.940430	8719.919922	8769.700195	8769.700195	266710000
4	2009-01-08	8769.940430	8770.019531	8651.190430	8742.459961	8742.459961	226620000

Fig. 1. A typical daily stock data: a sequences of float numbers representing an open and close prices, daily high and low, adjusted close price and the volume. Source: https://finance.yahoo.com

The next paragraphs present different approaches that researchers adopt to define an environment state, an action, and a reward.

A. State Space Definition

In the related work, the definition of an environment state vector varies and depends on the RL method. Some RL algorithms, such as tabular Q-learning, require discrete state representation and some can work with continuous states, such as DQN. Below is the list of several approaches from the most simple to more sophisticated ones.

- Take k last daily prices (typically adjusted close price) [3], [4]
- Take the current price along with the number of shares per stock. Add various technical indicators: moving average, etc. [2]
- Pre-process the raw data, so each record is tuple
  $$(\Delta Open, \Delta High, \Delta Low, \Delta Close, \Delta Volume)$$
  where each number corresponds to the percentage change compared to the previous day. Then, cluster the pre-processed data, so each cluster represents a state. This approach is based on the notion ”that history tends to repeat itself” [5]
- Define discrete states based on the open-close prices movements during the day [5].
- Use Gated Recurrent Unit (GRU) to derive a hidden vector state representation out of the raw data [6]

B. Action Space Definition

An automatic stock trading agent's natural choice of action is to *buy, sell, or hold*. However, variations such as *sell long, sell short, or hold*, are also possible [7] In portfolio management, for each action type, we decide the number shares: 1,2,..., *max*, an integer. However, to apply certain RL methods, such as Advanced Actor-Critic and Proximal Policy Optimization, the discrete action space can be normalized into continuous [2].

C. Reward Definition

Below is a list of possible reward functions that researchers adopt to formulate an RL problem on a stock market data.

- Reward is a relative change in close price compared to the open price:

$$R = \frac{p_{close} - p_{open}}{p_{open}}$$

  or zero if the action is *hold* [7].
- Reward is a difference between the current and the next day prices:

$$R = (p_{t+1} - p_t) \times a$$

  where a is an action from a set of *[-1,0,1]* [4]
- Reward is difference in the total balance after and before an action [2].
- Reward is a Sortino ratio [6] which is a statistical measure of a portfolio return with respect to a risk-free investment.

D. RL Algorithms

To tackle automatic stock trading, researchers adopt a variety of RL algorithms. The most recent approaches include, but not limited to, the following:

1) Critic-Only Algorithms: (Chakole et al., 2021) utilize Q-learning with an emphasis on data pre-processing to determine discrete environment states. (Chen, Gao, 2019) compare Deep Q-network (DQN) and Deep Recurrent Q-network (DRQN), which can detect hidden patterns in sequenced data. (Carta et al., 2020) use an ensemble of many DQN agents. The idea is to train each DQN agent for a different number of epochs so that each agent represents a trader with a different amount of expertise.

2) Actor-Critic Algorithms: While a critic-only network estimates a value function and an actor network learns a policy, the coupled actor-critic network combines them both. The actor learns the optimal policy, and the critic estimates the value function to evaluate the policy.

(Wu et al., 2020) compare the critic-only Gated DQN with the actor-critic Gated Deterministic Policy Gradient while preceding them with a GRU that generates latent state vectors. (Yang et al., 2020), whose work is a baseline approach for this project, combine Advantage Actor-Critic (A2C), Proximal Policy Optimization (PPO), and Deep Deterministic Policy Gradient (DDPG) into an ensemble. They periodically evaluate all three agents and choose the best one for trading.

The next section describes in detail the RL problem statement in the baseline approach, along with the training and evaluation strategy and the chosen algorithms.

III. Baseline Approach

The ensemble method from (Yang et al., 2020) [2] is a basis for this project. The goal is to make a profit with a portfolio of 30 stocks that represent a DJIA index. Below is the detailed description of the baseline method: an RL problem formulation, algorithms, training, and evaluation strategy.

A. Environment

The environment state is a 181-dimensional vector that includes the current money balance, price, and amount per stock. Also, it includes several technical indicators calculated from the raw data.

$$\mathbf{S} = [b_t, \mathbf{p}_t, \mathbf{h}_t, \mathbf{M}_t, \mathbf{R}_t, \mathbf{C}_t, \mathbf{X}_t]$$

where

- b_t is a current money balance in the portfolio
- $\mathbf{p} = [p_1, \ldots, p_{30}]$ is the adjusted close price per stock
- $\mathbf{h} = [h_1, \ldots, h_{30}]$ is number of shares per stock
- $\mathbf{M} = [M_1, \ldots, M_{30}]$ moving average convergence divergence
- $\mathbf{R} = [R_1, \ldots, R_{30}]$ relative strength index
- $\mathbf{C} = [C_1, \ldots, C_{30}]$ commodity channel Index
- $\mathbf{X} = [X_1, \ldots, X_{30}]$ average directional index

When an agent interacts with an environment, at each step, for each stock, it can choose one from

$$-k, \ldots, -1, 0, 1, \ldots, k$$

Where:

- -k means sell k shares
- +k means buy k shares

However, to apply continuous action-space RL methods, the action space is later normalized to [-1,1].

The reward is a difference in a total portfolio value before and after an action:

$$R = Account\ value\ after\ action - \\ Account\ value\ before\ action$$

The baseline approach utilizes three different actor-critic methods: A2C, PPO, and DDPG. The bellow describes those methods.

B. Algorithms and Evaluation Method

The baseline approach uses an ensemble of A2C, PPO, and DDPG. The A2C algorithm is an actor-critic network where instead of the value function, the critic estimates the advantage function:

$$A(s, a) = Q(s, a) - V(s)$$

A2C is a synchronous version of A3C, Asynchronous Advantage Actor-Critic, introduced by (Mnih et al., 2016) [8]. So, A2C also uses the same agent's copies that learn independently. However, the synchronicity in A2C allows computationally effective on hardware that does not allow many parallelization [2].

PPO is an improvement of Trust Regions Policy Gradient (TRPG) [9]. Like TRPG, PPO uses Kullback–Leibler (KL) divergence to ensure the new policy does not move too far away from the old policy to achieve stability and reliability. The PPO incorporates the KL constraint into the objective function as a "clipping" term, so the algorithm is much simpler than TRPG. Like A2C, PPO uses multiple workers to explore the environment more efficiently.

DDPG, introduced by (Lillicrap et al., 2019) [10], is an improvement of Deep Policy Gradient (DPG). DDPG combines DQN as a critic network and DPG as an actor network. The authors of DDPG invented the algorithm specifically for the continuous action spaces. The baseline approach uses DDPG ”to encourage maximum investment return.”

The baseline approach trains the above three agents independently. After training, we evaluate each agent's performance and choose the best one based on a Sharpe ratio:

$$Sharpe\ ratio = \frac{R_p - R_f}{\sigma_p}$$

where R_p is a portfolio's return, R_f is a risk-free portfolio return, and σ_p is a standard deviation of the portfolio's excess return.

Sharpe ratio evaluates the portfolio return with regards to the risk. A higher Sharpe ratio is better.

C. Training and Validation Strategy

Fig. 2 illustrates the stock data training-validation-test split.

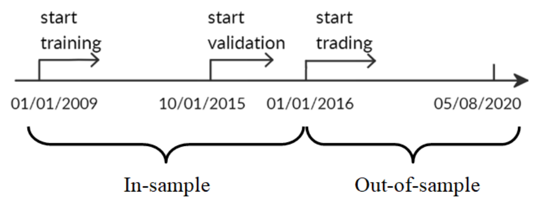


Fig. 2. Training and validation strategy. Source: [6]

Below is the description of the training, evaluation and testing strategy.

- Train each agent on the data from 01/01/2009 to 09/30/2015.
- Evaluate each agent on the data from 10/01/2015 to 12/31/2015 (3 month).
- Calculate the Sharpe ratio on validation data. For trading, choose the agent that yields the highest Sharpe ratio.
- Test (trade) on the data from 01/01/2016 to 05/08/2020.
- During trading, retrain and reevaluate on the new data every 3 month to choose the highest Sharpe ratio agent for future trading.

The next section describes the modifications that this work adopts to challenge the performance of the baseline approach.

IV. Modifications to the Baseline Approach and Results

A. Short State Vector

First, we check how well the 181-dimensional state vector is justified. We remove all the technical indicators, so the state vector now is only 61-dimensions:

$$\mathbf{S} = [b_t, \mathbf{p}_t, \mathbf{h}_t]$$

Fig. 3 shows the performance of both models on the test set. As we can see, the cumulative portfolio value, the account value over time, is higher for the original model with the 181-dimensional state vector.

B. MlpLstmPolicy

The baseline method's code uses the *stable-baselines* python library [11]. This library implements many RL algorithms. For actor and actor-critic methods, the *stable-baselines* provides several pre-defined policy networks.

The original method uses the default *MlpPolicy* for all three agents, A2C, PPO, and DDPG. *MlpPolicy* is ”a policy object that implements actor-critic policy using MLP (2 layers of 64),” as defined in *stable-baselines* documentation [12]. The MLP here stand for Multi-Layer Perceptron.

However, for A2C, we have an option to choose the *MlpLstmPolicy* - ”a policy object that implements actor-critic using LSTM with MLP feature extraction” [12]. LSTM, which stands for Long Short-Term Memory, is a construction that is suitable for ordered inputs sequences. Sequence to Sequence transduction neural networks often include LSTM cells. Since stock prices data is a sequence, we assume that *MlpLstmPolicy* might improve the A2C agent's performance.

Fig. 4 shows the performance of the original, short state vector, and *MlpLstmPolicy* policy model. The original model still performs the best on the test set, whereas the *MlpLstmPolicy* policy model is even worse than the short state vector model.

C. Turbulence Strategy

Financial *turbulence* is a market condition where stock prices move too extreme comparing to the historical price movements. The financial turbulence index is defined through the *Mahalanobis distance* that measures a distance of a given vector $\mathbf{y}$ from a sample of vectors with an average μ and covariance matrix Σ:

$$turbulence = (\mathbf{y} - \mu)\Sigma^{-1}(\mathbf{y} - \mu)'$$

where $\mathbf{y}$ is a vector of a portfolio returns for a given period and μ is an average portfolio returns vector based on the past data.

A proper trading strategy during turbulence allows risk aversion, safety of principle, and preservation of capital [13]. The baseline approach's strategy during turbulence is *sell all the assets and do not buy anything*. Here, we try two different strategies:

- *Hold (do not sell or buy anything)*

- *Hold and buy more*

Fig. 5 shows the trading system performance after adopting each of the above strategies. The original approach, *sell all the assets and do not buy anything*, performs better than the other two. The strategy *Hold (do not sell or buy anything)* appears to be the worst among the three.

D. Sortino Ratio

Like Sharpe ratio, the Sortino ratio measures the risk-adjusted portfolio return. However, in Sortino ratio, we divide by standard deviation of the negative return only.

$$Sortino\ ratio = \frac{R_p - R_f}{\sigma_p^+}$$

Thus, we don't penalize positive returns, like in Sharpe ratio.

In this experiment, we use the Sortino instead of the Sharpe ratio to evaluate the agent's performance on the validation set. Fig. 6 compares both trading systems. The original one that uses the Sharpe ratio performs better than the proposed one that uses the Sortino ratio.

E. Multi-PPO

The following experiment's idea comes from (Carta et al., 2020) [7] where authors train independent DQN agents for a different amount of epochs. In the baseline approach, the PPO agent alone performs better than the other two, A2C and DDPG. Thus, here we decide to experiment with multiple PPO agents trained for a different number of steps: 10000, 25000, and 50000 timesteps.

Fig. 7 compares the performance of the original ensemble trading system with the experimental one. The original trading system chooses the best agent among A2C, PPO, and DDPG, whereas the experimental one chooses the best among three PPO agents. The results show that the original ensemble is the better one.

Fig. 8 displays all the modified models along with the original one. All the modifications to the original trading system failed to improve it. The modified model with turbulence strategy *Hold (do not sell or buy anything)* is the worst among all presented. Fig. 9 compares the portfolio's total value over time, managed by the baseline trading system, with the DJIA index from 01/01/2016 to 05/08/2020. The graphs show that the original trading system beats the market.

V. Conclusion

In this project, the author studies an RL approach for automated portfolio management. We take an RL-based trading system that implements an ensemble of A2C, PPO, and DDPG learning agents for the baseline. We make several modifications to challenge the baseline approach: different environment state representation, evaluation metric, trading strategy under turbulence, and more. The results show that the original trading system performs the best, judging by the accumulated portfolio value. Future improvements may include the hyperparameters tuning and more frequent retraining and reevaluation during trading.

GitHub Link

https://github.com/LubaTovbin/CMPE-260

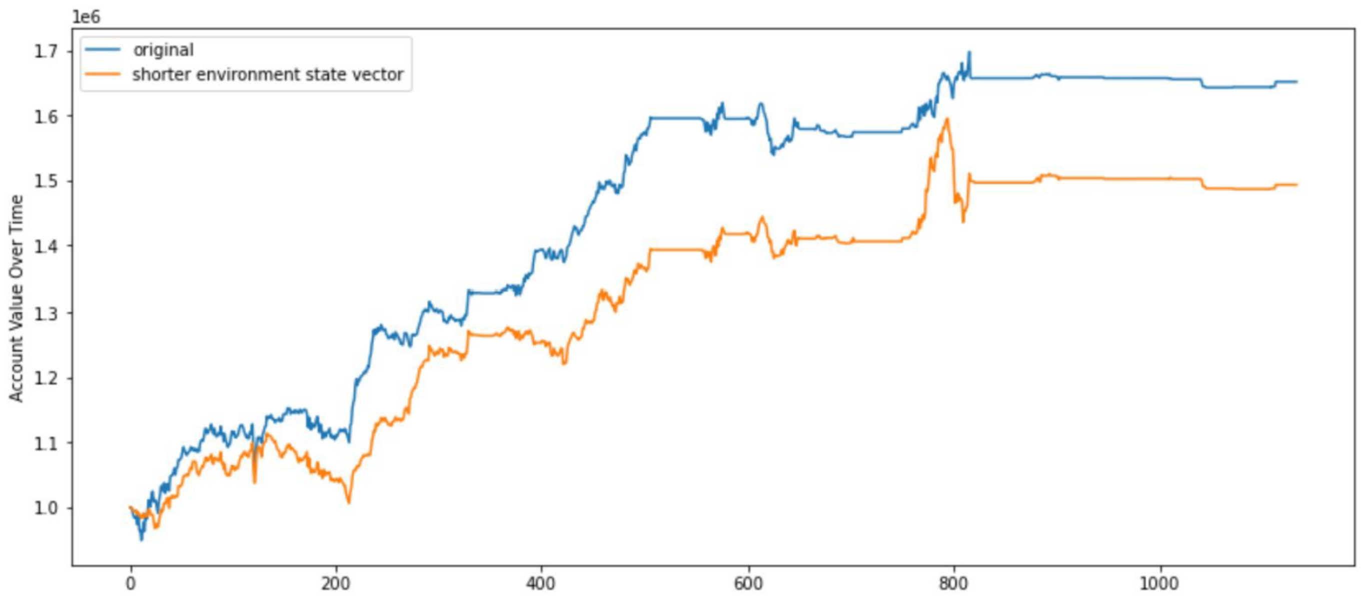


Fig. 3. The model's performance with the original 181-dimensional state vector (blue) and the modified one, with a 61-dimensional state vector (orange) without technical indicators. The x-axis represents days from 01/01/2016 to 05/08/2020.

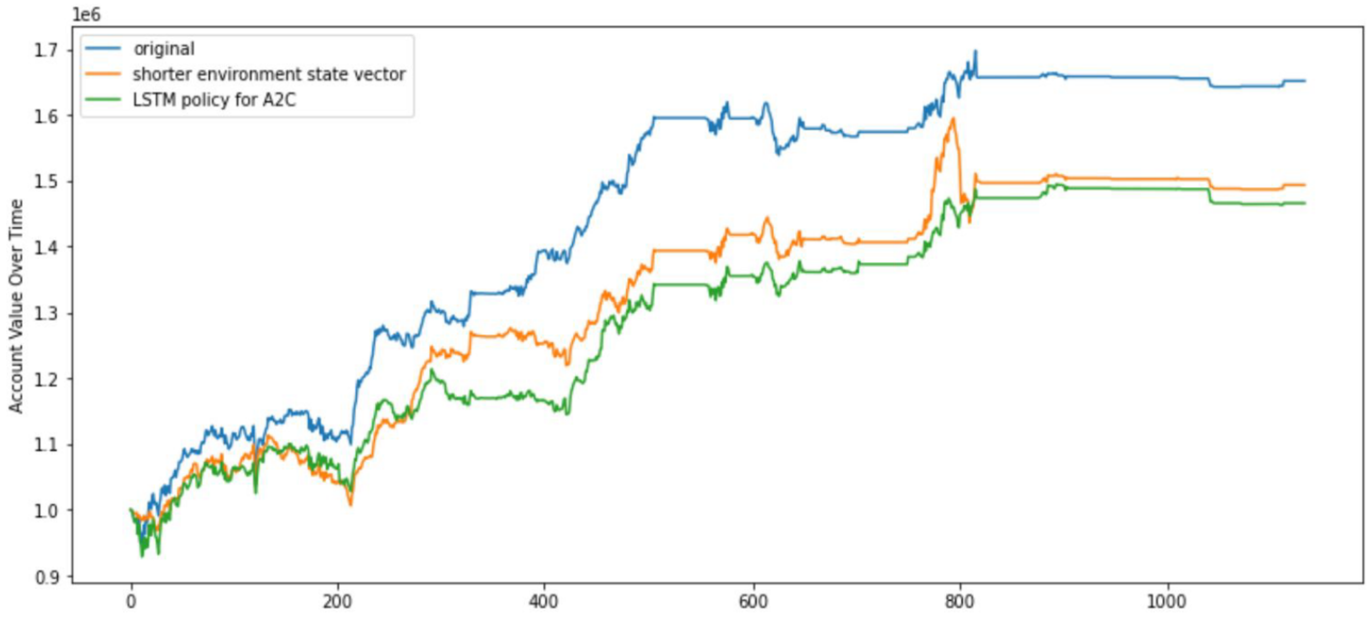


Fig. 4. The performance of the original, short state vector, and *MlpLstmPolicy* policy model. The original model (blue) performs the best on the test set, whereas the *MlpLstmPolicy* policy model (green) is even worse than the short state vector model (orange). The x-axis represents days from 01/01/2016 to 05/08/2020.

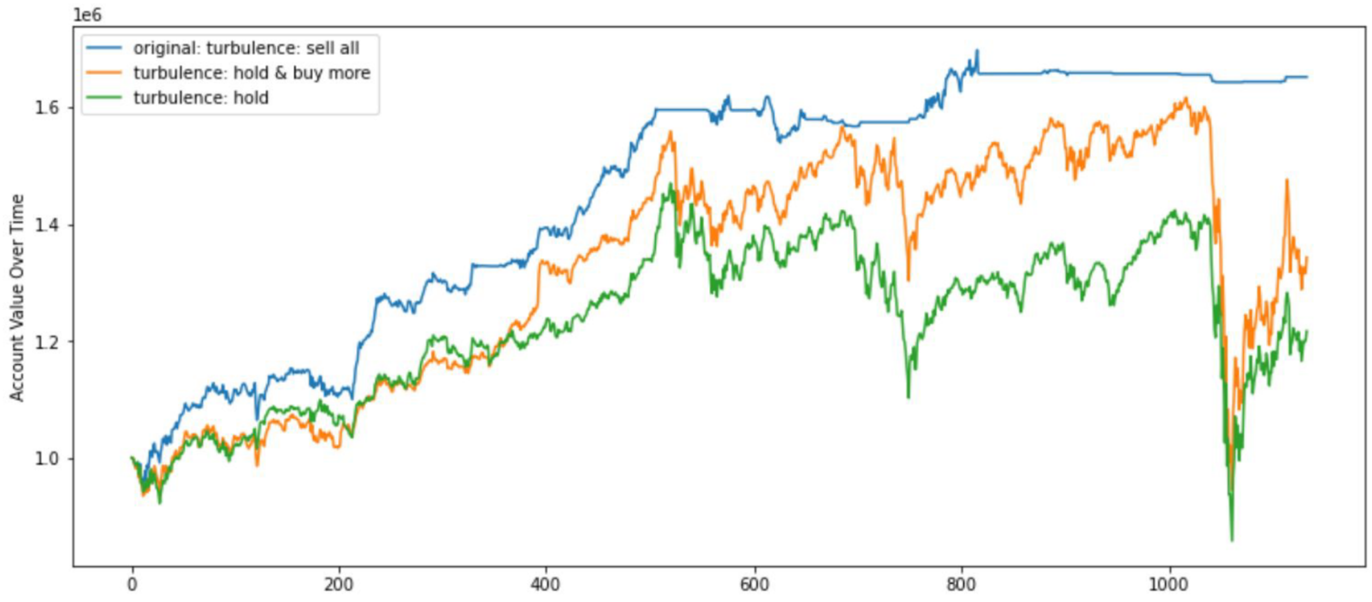


Fig. 5. The performance of the trading system after adopting different strategies during *turbulence*. The original approach, *sell all the assets and do not buy anything*, performs the best (blue). The strategy *Hold (do not sell or buy anything)* appears to be the worst (green). The x-axis represents days from 01/01/2016 to 05/08/2020.

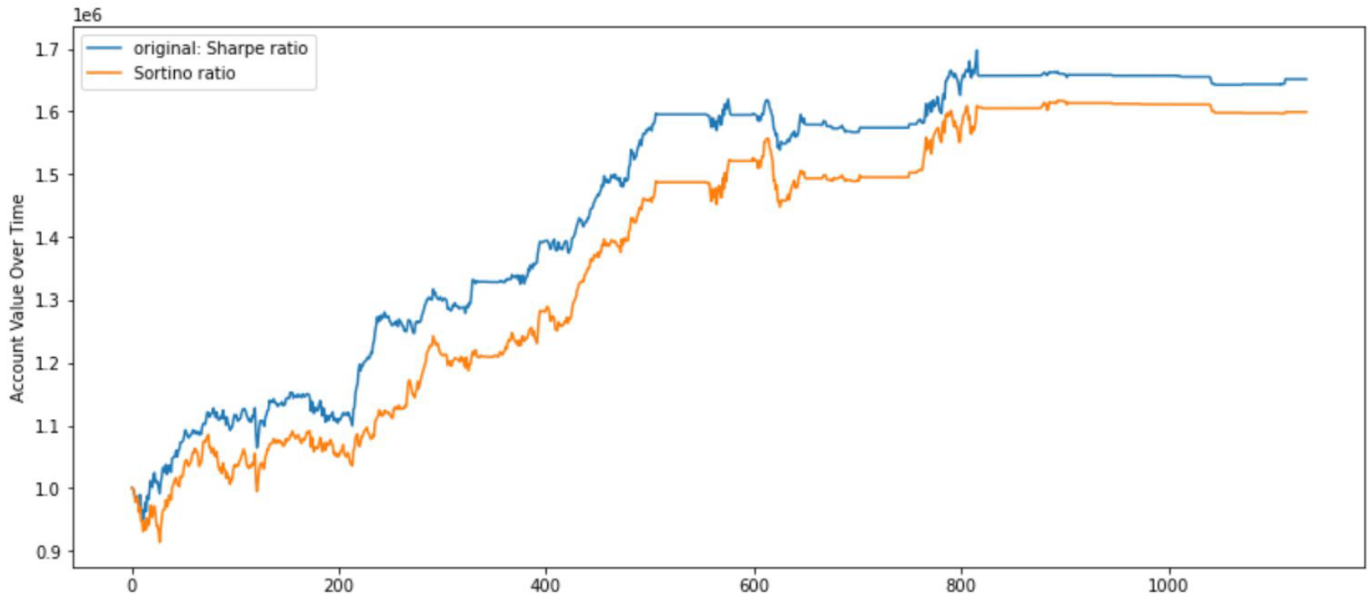


Fig. 6. The performance of the original trading system (blue) and the experimental one (orange) that uses Sortino instead of the Sharpe ratio to evaluate A2c, PPO, and DDPG agents. The x-axis represents days from 01/01/2016 to 05/08/2020.

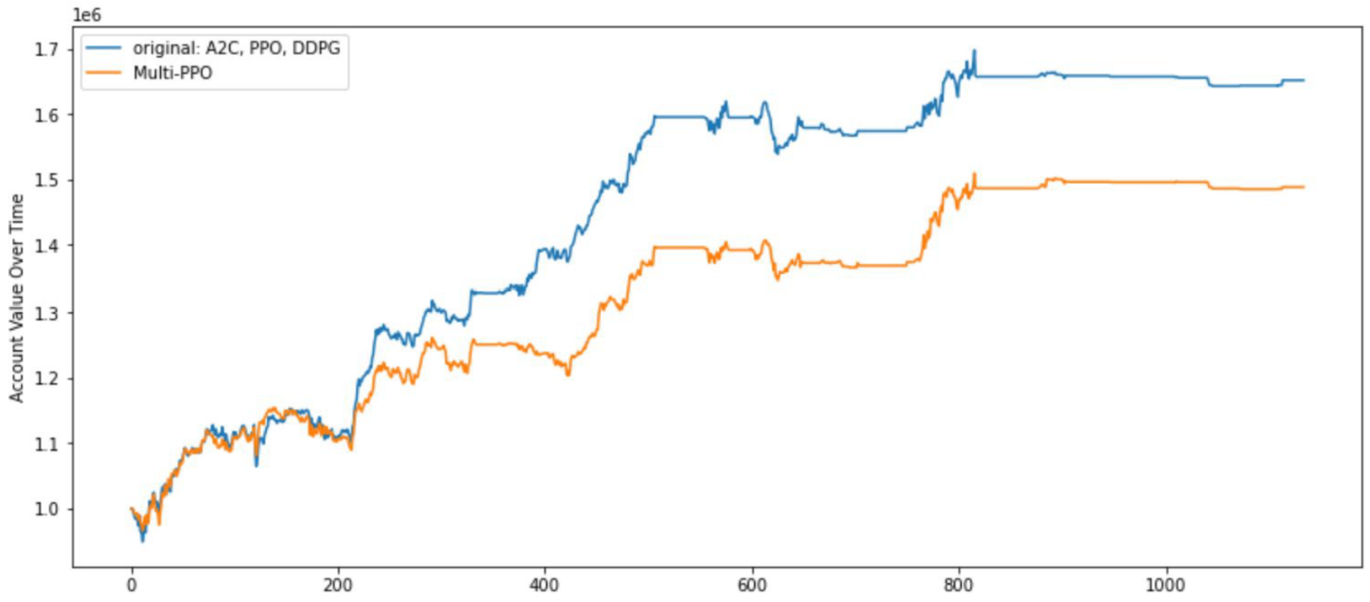


Fig. 7. The comparison of the original ensemble (blue) with the multi-PPO ensemble (orange). The x-axis represents days from 01/01/2016 to 05/08/2020.

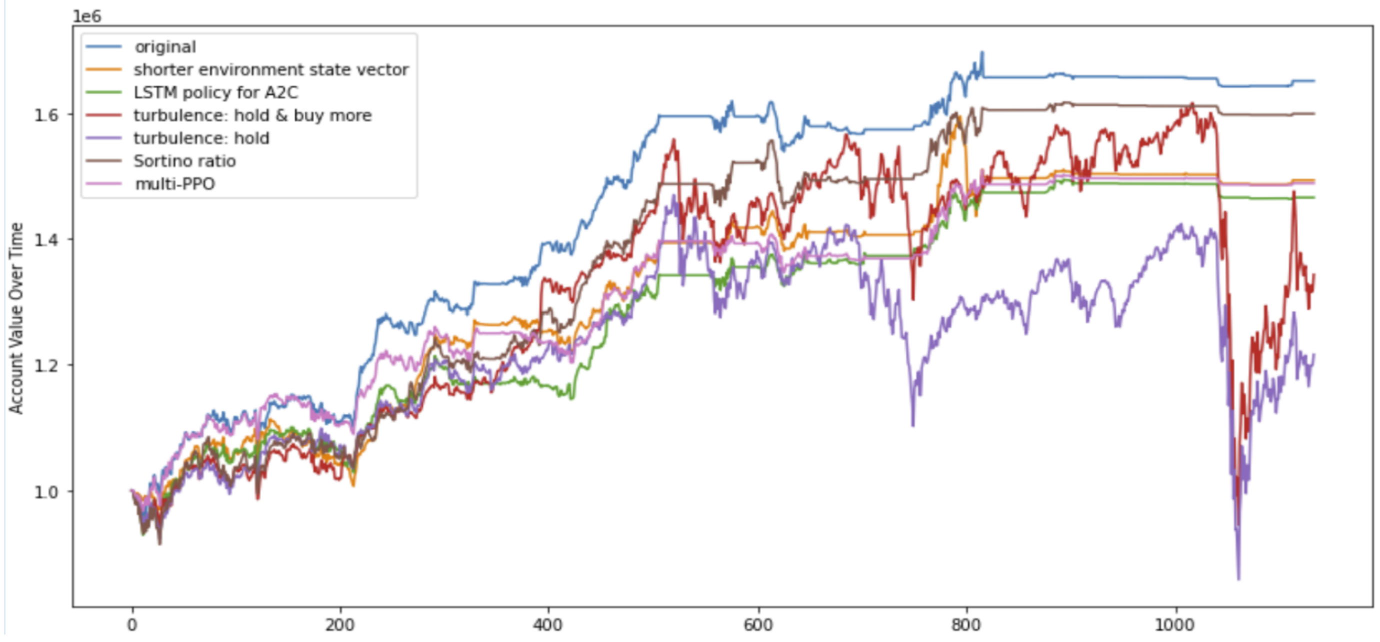


Fig. 8. The comparison of the original trading system (blue) with the modified ones. The x-axis represents days from 01/01/2016 to 05/08/2020.

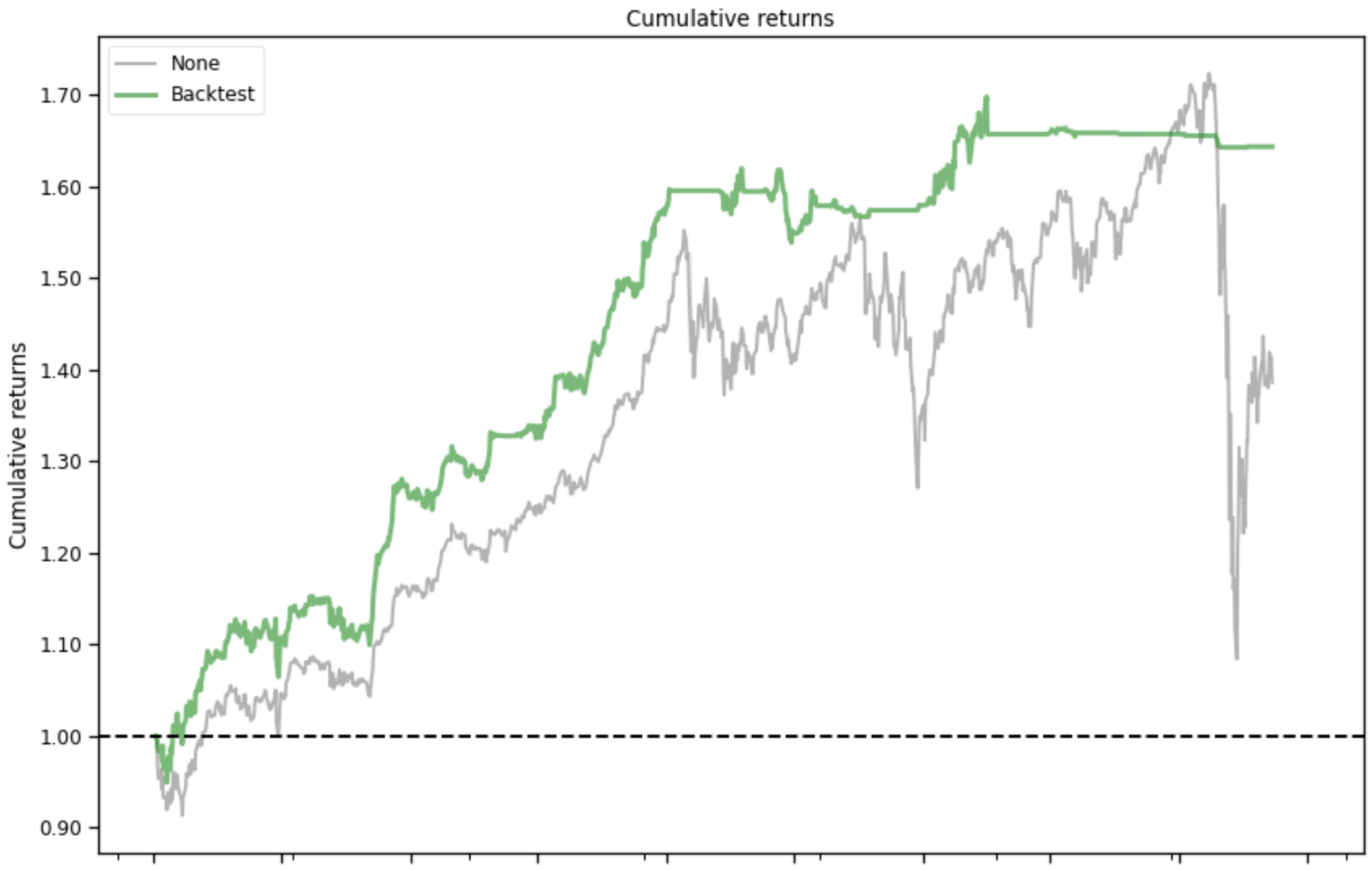


Fig. 9. The comparison of the original trading system (green) with the DJIA index (gray). The x-axis represents days from 01/01/2016 to 05/08/2020.

References

[1] Gandhmal, D., amp; Kumar, K. (2019, August 28). Systematic analysis and review of stock market prediction techniques. Retrieved December 07, 2020, from https://www.sciencedirect.com/science/article/pii/S157401371930084X

[2] Yang, H., Liu, X., Zhong, S., Walid, A. (2020, November 03). Deep Reinforcement Learning for Automated Stock Trading: An Ensemble Strategy. Retrieved December 01, 2020, from https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3690996

[3] Lapan, M. (2020). Deep Reinforcement Learning: Das umfassende Praxis-Handbuch. Frechen: Mitp.

[4] Chen, L., Gao, Q. (2019). Application of Deep Reinforcement Learning on Automated Stock Trading. 2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS). doi:10.1109/icsess47205.2019.9040728

[5] Chakole, J. B., Kolhe, M. S., Mahapurush, G. D., Yadav, A., Kurhekar, M. P. (2021). A Q-learning agent for automated trading in equity stock markets. Expert Systems with Applications, 163, 113761. doi:10.1016/j.eswa.2020.113761

[6] Wu, X., Chen, H., Wang, J., Troiano, L., Loia, V., Fujita, H. (2020, June 13). Adaptive stock trading strategies with deep reinforcement learning methods. Retrieved December 10, 2020, from https://www.sciencedirect.com/science/article/pii/S0020025520304692

[7] Carta, S., Ferreira, A., Podda, A., Recupero, D., Sanna, A. (2020, August 09). Multi-DQN: An ensemble of Deep Q-learning agents for stock market forecasting. Retrieved December 09, 2020, from https://www.sciencedirect.com/science/article/pii/S0957417420306321

[8] Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, Kavukcuoglu, K.. (2016). Asynchronous Methods for Deep Reinforcement Learning. Proceedings of The 33rd International Conference on Machine Learning, in PMLR¡/i¿ 48:1928-1937

[9] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O. (2017, August 28). Proximal Policy Optimization Algorithms. Retrieved December 04, 2020, from https://arxiv.org/abs/1707.06347

[10] Lillicrap, T., Hunt, J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Wierstra, D. (2019, July 05). Continuous control with deep reinforcement learning. Retrieved December 04, 2020, from https://arxiv.org/abs/1509.02971

[11] https://github.com/hill-a/stable-baselines

[12] https://stable-baselines.readthedocs.io/en/master/modules/policies.html

[13] Kritzman, M., amp; Li, Y. (2010). Skulls, Financial Turbulence, and Risk Management. Financial Analysts Journal, 66(5), 30-41. doi:10.2469/faj.v66.n5.3