Multi-agent Reinforcement Learning based Portfolio Management with a Confidence Generating System

Word Count: 4817

1. Introduction

The art of portfolio management has been around for as long as the stock market has existed. Portfolio management is the task in which an investor or an artificial intelligence continuously allocates various amounts of capital towards varying assets in their portfolio in an attempt to generate substantial profits. However, it was not until Markowitz and his publication in 1952 (Markowitz 1952) that portfolio management was thought of as a scientific, mathematical, optimization problem. Before then, portfolio management was conducted as a matter of sentiment and valuations based on company metrics. Years later, during the late twentieth century, technical indicators became extremely popular as a method to predict asset prices based largely on historical data alone. Technical indicators are not adaptive and therefore lack the ability to be continually successful in fast changing markets because they are simply mathematical formulas and fixed algorithms. Examples of technical indicators primarily include moving averages of the asset price and price ranges, which are used to give indications of future stock movement based on past trends.

A more recent form of portfolio management that has gained popularity is one that uses machine learning. The use of machine learning generally has grown immensely throughout the past decades due to its success in image recognition and many other fields. One niche of machine learning, called deep reinforcement learning, has witnessed tremendous growth in areas including robotics and gaming, and has applications in portfolio management. Deep reinforcement learning is different from classical machine learning in that an agent, which is typically a neural network, interacts with the environment as it learns. A neural network is a crude approximation of the human brain in an algorithmic format; however, similar to a brain, it is capable of learning. An example of this would be reinforcement learning's application to playing games, such as the Atari game Breakout. The reinforcement learning agent plays the game from the player's perspective, and it attempts to maximize the points received by improving via a reinforcement feedback loop.

1.1 *Background*

Recently, reinforcement learning has gained significant traction in the field of portfolio management due to its natural suitability for the problem. Deep reinforcement learning is naturally suited for the task of portfolio management because of how each action given by an agent (i.e., a neural network) directly impacts the return of the portfolio (the reward). However, simply throwing a neural network at the problem is not an efficient way to approach the problem. There have been many different approaches to this problem: single agent to multiagent systems, altered data formats, and even innovations in the neural network framework itself. Single agent systems are those that contain only one agent, and therefore only one neural network. Whereas multiple agent systems contain multiple agents, and therefore multiple neural networks, and they are generally more powerful than single agent systems and better at portfolio management. Altered data formats, called data tensors, simply refers to how the data is processed before it is fed as an input into the neural network. Innovations to the neural network itself most directly refers to the Ensemble of Identical Independent Evaluators (EIIE) architecture (Jiang et al. 2017).

One implementation of multiple agents in reinforcement learning utilized a technique known as ensemble learning (Lee et al. 2020), while another used stock specific agents to generate trading signals for the portfolio management agent (Huang & Tanaka 2022). The former consists of many agents that perform portfolio management on the same stocks, and then their sub-portfolios combine together to form one large portfolio. This is useful because it exploits the “wisdom of the masses” effect. In the latter, which created the current state-of-the-art model, stock specific non-portfolio management agents are created with the task of supplying the portfolio management agent with high quality signals to act on (buy, skip, or close). Other papers have focused more on altering the input, such as adding noise (small amounts of randomness) to the data (Liang et al. 2018), adding news sentiment data and a closing price prediction (Ye et al. 2020), or even a specific way to normalize the data (“squish” the prices to a number centered around one) (Jiang et al. 2017). Models generally benefit from quality data processing because better data provides the opportunity for better learning and better performance.

1.2 *Research Gap*

Throughout my review of the literature I identified several gaps in the research that presented new opportunities for experimentation. One such example for exploration is a system of confidence signal generating agents that can be combined with an agent for portfolio management. A confidence signal refers to a value - between one and negative one - that represents the likelihood that the stock will go up (positive) or down (negative). In the multi-agent reinforcement learning-based system for portfolio management (MSPM), (Huang & Tanaka 2022), the Evolving Agent Module (EAM) generates a signal (buy, skip, or close) for the Strategic Agent Module (SAM). The Strategic Agent Module is still left to not only determine how well each stock will do, but also how to synthesize each stock’s predicted performance with each other. In order to ease the strain of multitasking on the portfolio management agent, a confidence generating agent could be introduced, so that the only thing the portfolio management agent has to do is compile the confidence signals among the assets into a portfolio.

Another area for exploration is the idea that no piece of literature in this field has used two or more non-portfolio management that provide signals based on various forms of data for the portfolio management agent. The MSPM, for example, only possessed one non-portfolio management agent: the EAM.

My research goal, using the model I created, was to beat the state-of-the-art reinforcement learning based portfolio management system.

2. Methods

2.1 *Problem Definition*

In portfolio management, the goal is to continuously reallocate capital into any number of assets to maximize return. As Markowitz stressed, however, we should seek not only to

maximize returns, but also simultaneously minimize volatility within a portfolio (Markowitz 1952). Not including the past few decades, portfolio management was conducted only by humans. Now, the process of portfolio management has been automated and is largely performed by computers and algorithms. For this reason, my research used only quantitative methods. Since the research is conducted through coding and mathematics, every aspect of it can be, and is, quantifiable.

The following subsections discuss the data collection and transformation, trading information, and the multi-agent framework involved in the portfolio management.

2.2 *Data*

For data, I imported stock data from the Dow Jones Index. I picked three stocks with the highest average volume that were also in different industries. It is important that these stocks are in different industries so that the correlation in price movements among them is not high. I chose Microsoft, Disney, and Johnson & Johnson to be the stocks in my research. The data extends from the beginning of 2000 to the end of 2020, which is twenty-one years of market data. The types of data included are the daily high, low, close, and volume data, as well as fundamental data such as P/E (price to earnings ratio). I omitted open prices for my model to eliminate additional complexity in the data input. Also, I included fundamental data in my model to provide more stock specific information. I also imported three month LIBOR interest rate data for use in training as discussed later

The data (which is also referred to as a price vector) goes through several modifications before it is inputted into the neural network. First, a log transform is applied to the close, high, low, and volume features. Applying log to price data can make the data conform to a more normal distribution which is beneficial to a neural network. Also, the volume data points are very large numbers, so undergoing a log-based scale for the volume data helps increase convergence speed because it makes those numbers much closer not only to the other data points but also closer to zero. Then, a small amount of noise generated from a normal distribution - for each stock i - is added to the close, high, and low features.

$$\text{noise}_i = N(0,0.1) * \sigma_i * 0.1$$

Note that the standard deviation calculated for each stock i.

After that, the close, high, and low features are normalized by dividing each by the last close price in the window. The volume feature is also normalized in the same fashion. Before the price vector is input into the neural network, a 10 day moving average (Hull moving average) is calculated from the modified close prices and added to the price vector. This is done to provide a smoothed version of the now noisier price action to the neural network which theoretically boosts performance. The shape of the price vector is (9,4,63). The first number is the height and the second is the width. The third number denotes the number of channels in the convolutional neural network (CNN). This is rather unorthodox since the first number is usually the number of channels, followed by the height and width. The reason behind this abnormal price vector shape is this: I thought that it would be more beneficial to the model's predictive

ability for it to prioritize learning about each feature's predictive capacity across each asset than vice versa.

After the data was processed as described above, the neural network is fed a rolling window of sixty three days of data. I chose not to go with the standard fifty day rolling window and instead with a sixty three day rolling window because that is the length of a quarter. There are approximately two-hundred fifty two trading days in a calendar year, which means that the number of trading days in a quarter is approximately sixty-three.

2.3 *Trading Information*

Since the stocks chosen were – on average – very voluminous, the assumption is that zero slippage occurs during trading. Slippage simply refers to the difference between the price that the market order was requested and the price that the order was executed at. With the same logic, the simulated trades also have no market impact. With each trade, however, there is a 0.25% transaction cost that goes into the return calculation.

2.4 *State-of-the-Art Framework*

Based on the general information from the research paper (Huang & Tanaka 2022), I programmed the state-of-the-art model so that I could test it in the environment I created.

As mentioned above there are two types of agents in MSPM. The first is the Evolving Agent Module (EAM), and the second is the Strategic Agent Module (SAM). There is one EAM for each stock. The buy, skip, and close signals for each stock and from each EAM in the portfolio are compiled and given to the SAM. The SAM simply predicts the percent allocations for each asset in the portfolio that will result in the largest return. The EAM is fed both daily price data, and fundamental data. However, as with my model, the SAM is fed the daily price data and LIBOR interest rates.

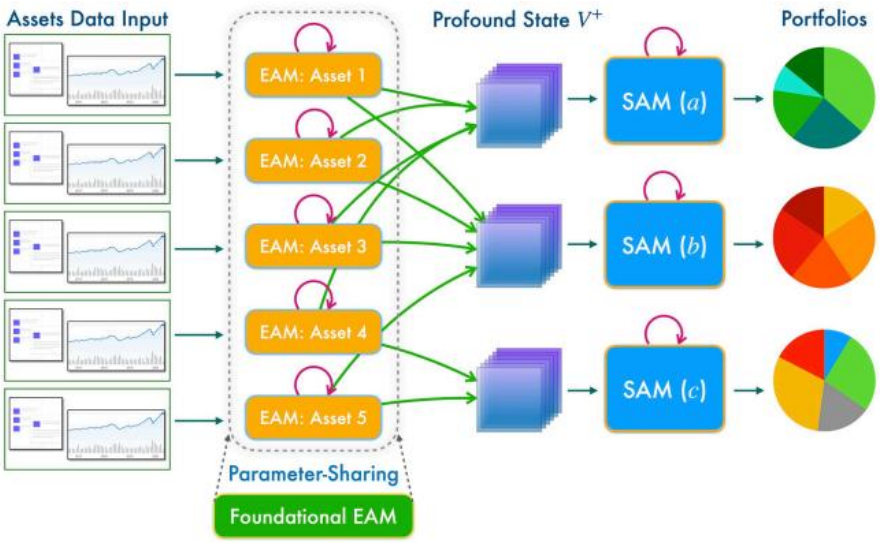

Figure 1: Diagram showing the model structure of MSPM (Huang & Tanaka 2022).

2.5 *My Multi-agent Confidence Generating Portfolio Management System*

There are three types of agents in this system I developed: signal generating agent (SGA), confidence generating agent (CGA), and a portfolio management agent (PMA). A functional diagram of my model is provided below in Figure 2.

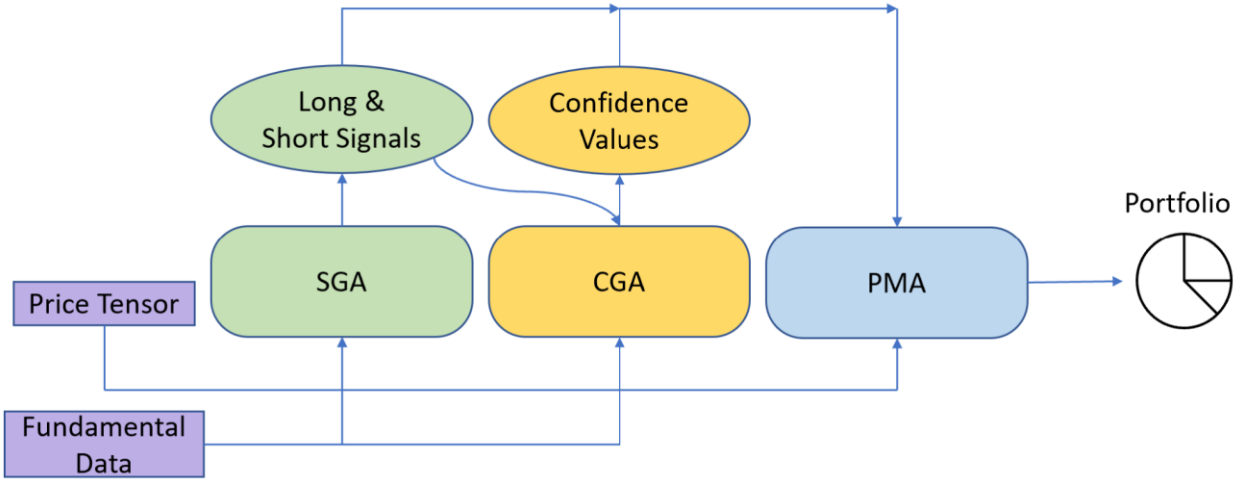

Figure 2: Diagram of my model.

The SGA outputs long and short signals. The CGA outputs a scalar from one to negative one corresponding to how long or how short the PMA should invest in the stock. The CGA output could also be interpreted as a confidence scalar (hence the name). The PMA then takes the two signals as additional inputs and allocates certain percentages of the portfolio to the assets. The idea is that the SGA generates the side of the trade (i.e., buy or sell), the CGA determines how much, and the PMA uses both to output allocation weights.

The algorithm used in SGA is a Deep Q Network (DQN), which is an off-policy method for calculating gradients to train a reinforcement learning agent. The extensions I used with DQN are two-step Bellman unrolling, as well as the Double and Dueling versions of the basic DQN. Instead of employing an e-greedy policy for exploration - which is the basic method of exploration for DQNs, I created a NoisyDense layer (Fortunato et al. 2017) and added it toward the end of the neural network so that it may undergo “self-guided” exploration. I did this because e-greedy has been found to be insufficient for complex environments (Fortunato et al. 2017). The reward function used for the SGA is called differential Sharpe ratio (Moody & Saffell 2001). The differential Sharpe ratio is a way to account for the change in the original Sharpe ratio, and has been found to be very effective as a reward function for agents that only provide long and short predictions (Moody & Saffell 2001). The parameter used in the differential Sharpe ratio that accounts for the amount of change each time step affects the function was set to 0.1.

The algorithm used in CGA and PMA is called Proximal Policy Optimization (PPO), which is an on-policy method for calculating gradients and training a reinforcement learning agent. As with the SGA, I also implemented NoisyDense layers into the neural network. Unlike the SGA, however, I also added an entropy term to the loss function to encourage further

exploration. The entropy term simply adds a small amount of randomness to the loss function. In the PMA, only the entropy term is used to aid exploration. The reward function used for CGA was simply a log of the return minus the transaction cost. The reward function used for PMA was similar to the CGA, but there is a log volatility term subtracted from the log return. This was done to account for the portfolio volatility, which should be minimized as the return is maximized.

At the center of the SGA and CGA is a 1-D convolution ResNet as the neural network responsible for the sequential decision making. ResNets are powerful neural network architectures that primarily utilize convolutional layers in the form of residual units. ResNets have been shown to perform extremely well on image recognition tasks, which demonstrates its ability to recognize patterns (He et al. 2015). Specifically, the ResNet type used in this paper is ResNet-34.

For the PMA, I deviated from using the EIIE neural network architecture proposed in "A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem" (Jiang et al. 2017). Instead, I created an architecture that contained separate data streams not for each individual asset, but for each feature in the price vector. This was done based on the idea that it would be more beneficial to focus on the predictive value of a feature across each asset rather than the value of a stock across each feature. For ease of replication, below are links to the code of each agent respectively:

- My model's SGA code here: https://tinyurl.com/SGAcode
- My model's CGA code here: https://tinyurl.com/CGAcode
- My model's PMA code here: https://tinyurl.com/4tk7apze

3. Experiment

In this section, I describe the portfolio used in training and testing of both the state-of-the-art and my model, as well as the various data ranges for training and testing. After that, I explain some of the hyperparameters of the agents and the trading environment that were used during the experiment. Then, I provide definitions for the metrics used to evaluate the performances of PMA and the baseline comparisons. A baseline is a nominal strategy in portfolio management that models are compared to in order to gauge performance.

3.1 *Portfolio*

In this experiment only one portfolio was constructed and used to test my model against the baselines. The portfolio consists of the three stocks Microsoft, Disney, and Johnson & Johnson (NYSE tickers: MSFT, DIS, and JNJ). The initial portfolio value is set to ten-thousand U.S. dollars in cash.

3.2 *Data Ranges*

There are three groups of data used, training data, validation data, and test data. The training data is simply used to train an agent. Validation data is not data that the agent trains on;

it is instead used to test or validate an agent's performance during training without having to use the test data set to gauge learning progress. Test data is the data that the performances of each model is tested on. For these reasons, the training data, validation data, and test data do not overlap date ranges.

Since only one SGA and one CGA is used generally to train on every asset in the portfolio, each time period is the same among each asset in the portfolio. The SGA is trained first, so that its signals may be passed onto the CGA for training purposes. In table 1, the SGA training data set ranges are shown, and these data sets contain not only historical prices (p_t) but also corporate fundamental data. The SGA predictions for each asset are then extracted and appended to the training data for the CGA, as well as the validation set and the test set to be used as signal-comprised data. The CGA is then trained in the period shown in the table below with historical prices (p_t), corporate fundamental data, and the signal-comprised data. The signals generated by the CGA are appended onto the signal-comprised data to be used for PMA training, validation, and testing. Unlike the two agents before it, however, the PMA is trained only using historical prices (p_t) and signal-composed data. Instead of using corporate fundamental data, I exchanged that for LIBOR data, which is a macroeconomic indicator consisting of interest rates. In this experiment I chose a three-month interest rate. The three datasets that include complete signal-composed data are PMA training, PMA validation, and PMA training to train, validate, and test the PMA, respectively.

Purpose	**Data Range**
SGA-training	Jan 2000 ~ Dec 2016
SGA-validation	Jan 2017 ~ Dec 2018
SGA-prediction	Jan 2019 ~ Dec 2020
CGA-training	Apr 2000 ~ Dec 2016
CGA-validation	Jan 2017 ~ Dec 2018
CGA-prediction	Jan 2019 ~ Dec 2020
PMA-training	Jan 2011 ~ Dec 2016
PMA-validation	Jan 2017 ~ Dec 2018
PMA-experiment	Jan 2019 ~ Dec 2020

Table 1: Data ranges for each data set.

3.3 *Environment*

Several assumptions were used for the trading environment in this experiment.

1. Market orders are executed immediately and without slippage

2. Market orders do not impact the volume or price of the asset

When market trading volumes are large, these assumptions approximate an actual market. Also, since the starting portfolio value is ten-thousand dollars, trading this portfolio would not be able to affect the volume or price of the high-volume stocks selected. However, there is a transaction cost of 0.25% applied to every transaction in the environment.

3.4 *Performance Metrics*

To measure the performance of the PMA and the baselines, I chose the following metrics. The first of which is one of the most basic measurements of success in portfolio management: annual rate of return (ARR). The annual rate of return is

$$\text{ARR} = (p_T/p_0\ )^{252/T}$$

where p_T is the price at the terminal time step, meaning the last data point that was used, and p_0 is the first price.

The reason ARR is better than the simple accumulative portfolio value in this case is because it is unfair to compare two portfolios in which management started at different times. However, the annual rate of return alone is not a sufficient performance metric because it does not measure the risk undergone by the portfolio. Due to the lack of representation of risk in the portfolio, I decided to include the Sharpe ratio (SR) (Sharpe 1964, 1994). The Sharpe ratio is a risk-adjusted average return. The Sharpe ratio is mathematically defined as

$$\text{SR} = E[p_t - p_F]/\sigma$$

where p_t are the daily returns, and p_F is the rate of return of the risk-free asset. Since the risk-free asset in this experiment is cash, the risk-free return is zero, $p_F = 0$. The numerator of that equation represents the expected or mean return, which is then divided by the standard deviation of the returns of the portfolio.

4. Results

In order to determine the viability of my model, I tested its performance and compared it with other proven strategies. Included in the testing are three other baselines, two traditional and the state-of-the-art (SOTA), as described below:

- (Uniform) Constantly Rebalanced Portfolio (CRP) strategy where at every time step the portfolio is uniformly rebalanced according to how many assets reside in the portfolio. It has been shown to be difficult to beat (DeMiguel et al. 2007).
- Buy and Hold (BAH) strategy which simply involves investing equally among assets at the first time step without further rebalancing.
- MSPM refers to the multi-agent reinforcement learning based system for portfolio management (Huang & Tanaka 2022), which is the current SOTA in reinforcement learning based portfolio management.

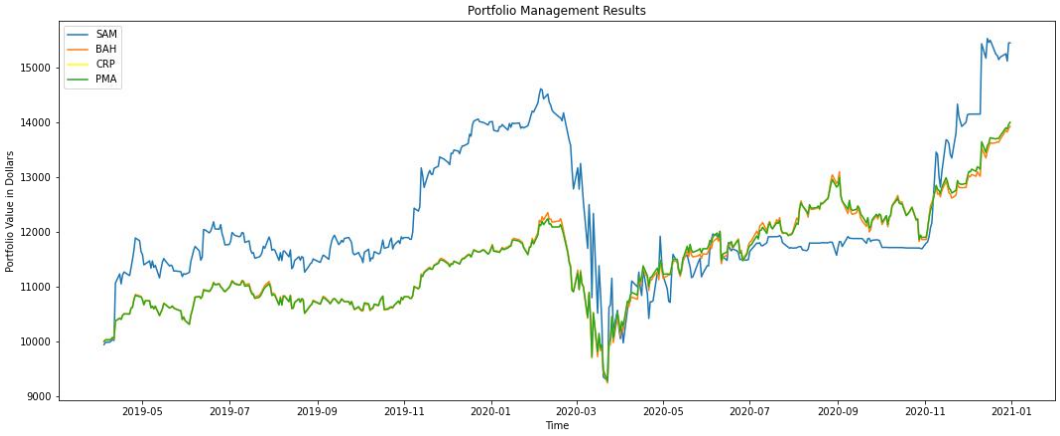

Figure 3: Portfolio values over time for each portfolio.

Metric	ARR (%)	Sharpe ratio
MSPM	**28.2**	0.450
My model	21.2	**0.754**
CRP	21.2	**0.754**
BAH	20.8	0.723

Table 2: Comparison of performance of the baselines to my model.

The ARR is calculated by cumulatively multiplying the daily rate of returns for a year. The ARR is a good measurement of return because it determines the return or profit that would have been gained in a year. The Sharpe ratio is a measure of the risk-adjusted return. A higher Sharpe ratio is deemed better because that means the portfolio has a comparatively higher return or comparatively lower risk. The Sharpe ratio is often used by organizations and funds to track their performance.

As shown in the table above, MSPM outperformed my model by seven percent for the time period modeled. Despite MSPM's ARR, its Sharpe ratio was significantly lower than my model. This is most likely due to the sharp downturn during COVID-19. The reason my model and the CRP both possess the same ARR and Sharpe ratios is because my model trained itself to give the actions of a CRP. This is also why the yellow line of the CRP does not appear on the graph. The PMA learned - rather suboptimally - that CRP is the best strategy. While CRP is a good strategy, it has been shown to be beatable. BAH's performance closely resembles that of the CRP. This is likely because both strategies closely follow the market and no individual stock swayed significantly compared to the others.

To better understand the SGA and CGA contributions to PMA, I illustrated their signal and confidence outputs for the test period and provide them in Figures 4-6 below. The figures represent the stocks used in this experiment: MSFT, DIS, and JNJ. In each graph, there are two subplots. The first shows the output of the SGA in an overlay of the long and short signals onto the stock's closing price line. Green represents long signals, and red represents short signals. The second plot contains the confidence values outputted by the CGA. The outputs can range from negative one to positive one depending on the CGA's "confidence" in the stock's future.

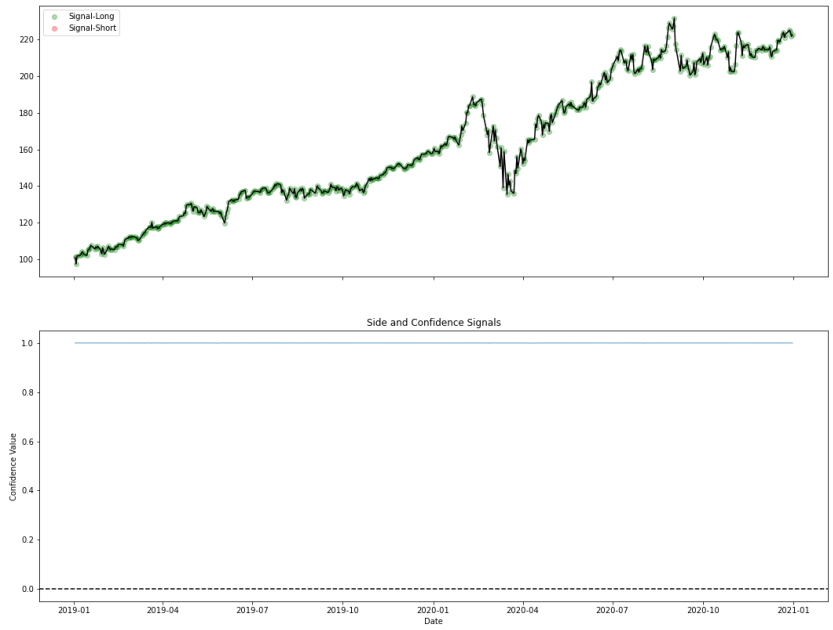

Figure 4: Signals and Confidence values of SGA and CGA on MSFT.

Figure 5: Signals and Confidence values of SGA and CGA on DIS.

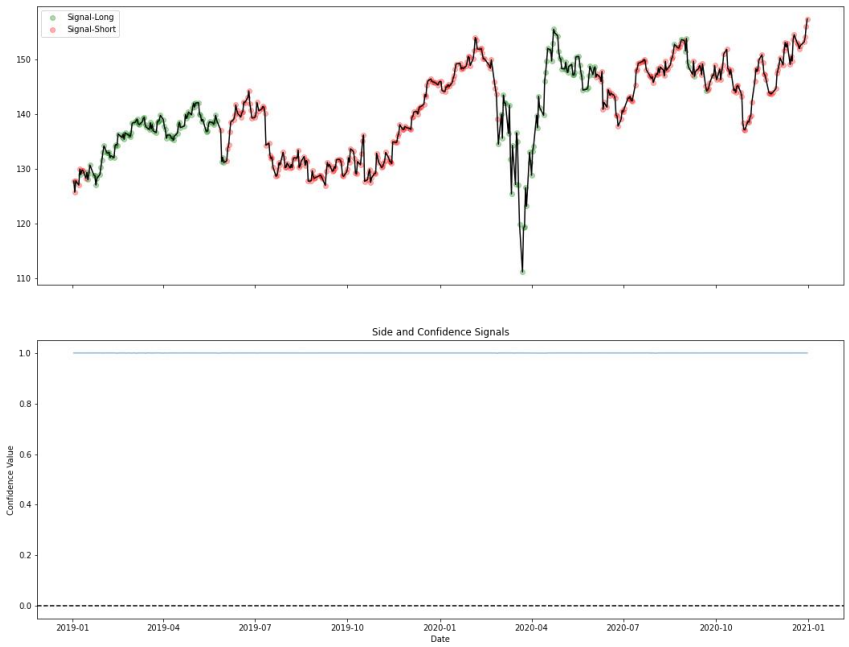

Figure 6: Signals and Confidence values of SGA and CGA on JNJ.

It should be noted that every confidence value outputted by the CGA is at or close to one for each of the stocks above. This is likely due to an extreme over or under fit of the training data by the CGA. This likely contributed to my model's failure to beat the SOTA. The SGA's behavior, however, is much more interesting and erratic than that of the CGA. The SGA does rather intuitively well on DIS, where it only shorts near and during the COVID-19 event. The SGA did not do as well on MSFT or on JNJ. This shortcoming also likely contributed to my model's failure to beat the SOTA. The suboptimal performance of the PMA could be partially attributed to these shortcomings of the SGA and the CGA.

5. Scalability and reusability of PMA

While my model is not directly reusable for other stocks, it is scalable and reusable subject to retraining. Since SGA and CGA were not stock specific, they are both not reusable if more assets were added to the portfolio. If the number or type of assets were changed the SGA and CGA would need to be changed as well because they only learned to generalize over the three stocks used in this experiment. However, it does not hinder scalability because each SGA and CGA would remain the same size. Instead, they would have to be retrained to generalize over an increased number of assets. The PMA is also relatively scalable because as the number of assets increases, the size of the PMA increases linearly. Naturally with every change in portfolio composition, the PMA would need to be retrained.

6. Conclusion

In this article, a new multi-agent and neural network framework was proposed for solving the portfolio management problem. The multi-agent approach is different from others in that it has two asset-dedicated agents. The first asset dedicated agent being the SGA, and the second being the CGA. The first utilizes a DQN-based agent to generate buy and short signal data for the next two agents. The second utilizes a PPO-based agent to generate confidence signal data for the next agent. This strategy was inspired by the multi-agent system MSPM, and it was meant to offload part of the task of portfolio management onto larger, more powerful asset dedicated agents. The SGA and CGA work together to provide an estimate of momentum - and therefore confidence - for how an asset will perform in the future. Theoretically then, all the PMA has to do is compile those signals from each asset and create a portfolio out of them.

One possible shortcoming discovered is that using non-stock specific SGA and CGA's likely decreased the predictive capabilities across each stock. That is because the agents had to learn to generalize across different companies in different sectors. Also, the lack of variability in the CGA's outputs rendered those signals with little informational and predictive value.

The SGA, CGA, and PMA were then trained and the my model was tested alongside baselines to determine the comparative effectiveness of the proposed multi-agent system. The results were measured by the annual rate of return and the Sharpe ratio. The annual rate of return of the state-of-the-art was significantly higher than my model and other baselines; however, its Sharpe ratio was lower than that of the PMA and therefore the CRP.

The new neural network framework developed in this research was designed to be an alternative to the EIIE framework. Both are easily scalable with the addition of more assets or more features. The significant difference between the two is that the neural network in the PMA prioritizes learning about each feature individually as it compares to the assets. Whereas, the EIIE prioritizes learning about each asset individually as it compares to the features. Since there are many factors that go into the performance of each model, it is impossible to rule out the viability of the framework within PMA. Similarly, it would be very difficult to conclude that the entirety of the multi-agent system proposed in this research does not hold value due to the many interrelated variables that affect performance. However, it is almost certain that the CGA's rather poor generalizability on the test set significantly impacted the PMA's performance. Also, the change to stock-specific SGA and CGA's would likely boost performance since there would be an agent for each stock instead of one agent that has to generalize over each stock.

Despite my model's failure to beat the ARR of the current state-of-the-art, my model showed that it is capable of beating the market (BAH), and that it had less volatility than the state-of-the-art. The results confirm the idea that a data-driven approach combined with machine learning - more specifically reinforcement learning - can optimize investment performance and beat the market.

7. Practical Implications

This research could be used in hedge funds and banks, particularly those that possess a quantitative trading related department. They would also be interested to know that the state of the art still maintains that status. As a result of my research, hedge funds and banks might be more inclined to create separate models for each task. Hedge funds and banks may also be inclined to implement asset-specific agents for each task. This idea was used in my research by the breaking up of the different steps into separate models (SGA, CGA, and PMA).

8. Implications for the Field of Research

There is much room for improvement in my model, and in this area of research in general. One way researchers could use my research is to incorporate an option to short within the PMA. This would mean that the PMA could better utilize the long and short signals given by the SGA and CGA for portfolio optimization. Another area for improvement would have been to include leverage into the portfolio. Leverage is a very commonly used tool in hedge funds and banks to significantly inflate returns by taking on additional risk. By including leverage into the portfolio management process, the model and research would be much more realistic, as well as more easily integrated into a real trading scenario. Also, to make the experiment more realistic, slippage could be accounted for within the simulation. There are still many things to experiment with when it comes to reinforcement learning based portfolio management.

Bibliography

DeMiguel, V., Garlappi, L., & Uppal, R. (2007). Optimal Versus Naive Diversification: How Inefficient is the 1/N Portfolio Strategy? *The Review of Financial Studies.* 22(5):1915–1953. https://doi.org/10.1093/rfs/hhm075

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. *arXiv.* https://doi.org/10.48550/arXiv.1512.03385

Huang, Z. & Tanaka, F. (2022). MSPM: A modularized and scalable multi-agent reinforcement learning-based system for financial portfolio management. *PLOS ONE, 17*(2): e0263689. https://doi.org/10.1371/journal.pone.0263689

Jiang, Z., Xu, D., & Liang, J. (2017). A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem. *arXiv*. https://doi.org/10.48550/arXiv.1706.10059

Lapan M. (2018). Deep Reinforcement Learning Hands-On: Apply Modern RL Methods, with Deep Q-Networks, Value Iteration, Policy Gradients, TRPO, AlphaGo Zero and More. *Packt Publishing*.

Lee, J., Kim, R., Yi, S.-W., & Kang, J. (2020). MAPS: Multi-Agent reinforcement learning-based Portfolio management System. *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence,* 4520-4526. https://doi.org/10.24963/ijcai.2020/623

Liang, Z., Chen, H., Zhu, J., Jiang, K., & Li, Y. (2018). Adversarial Deep Reinforcement Learning in Portfolio Management. *arXiv.* https://doi.org/10.48550/arXiv.1808.09940

Markowitz, H. (1952). Portfolio Selection. *The Journal of Finance*, *7*(1), 77–91. https://doi.org/10.2307/2975974

Moody, J., & Saffell, M. (2001). Learning to trade via direct reinforcement. IEEE transactions on neural Networks 12(4):875– 889. https://doi.org/10.1109/72.935097

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. *arXiv*. https://doi.org/10.48550/arXiv.1707.06347