Stock Portfolio Optimization with Machine Learning - an ECE228 Project (Track Number # 1)

Group Number # 23: Yalu, Ouyang
yaouyang@ucsd.edu

Darell, Chua
dchuayun@ucsd.edu

Dirk, Xie
six004@ucsd.edu

Kyle, Wade
kwade@ucsd.edu

Abstract

Our project seeks to use machine learning techniques to tackle the problem of stock portfolio optimization. We use stock market prices obtained from publicly available data to compile a sample portfolio on which we conducted experiments. In addition to Neural Network architectures covered in ECE 228, we also explored other ML techniques to predict stock prices and optimize portfolios accordingly. The results of our experiments were compared to a baseline index to gauge relative performance on investment valuation growth. The corresponding code for our project can be found at `https://github.com/kyle1373/ece228-project`.

I certify that I have filled out the evaluation.
- all group members

1 Introduction

Wealth management is an important and complex topic. It is in the interest of everyone to ensure that their assets become more valuable, or at the very least don't depreciate about inflation. But because this topic is incredibly complex, with many sub-fields dedicated to it, it becomes extremely difficult for an average person to obtain a holistic understanding of what they need to do to maximize return on their investments. Thus we focused on one such sub-field, stock portfolio optimization, and explored ways of obtaining better performance using machine learning techniques.

The school (UC San Diego) holds a stock portfolio that is advised by a committee of students. We wish to develop an application that can optimize the constituents of this particular stock portfolio to maximize return on investment. The goal is not to change the constituents of this portfolio by adding and/or removing stocks; rather, it is to optimize the weight of each stock to obtain the biggest increase in valuation, and thus, return on investment. Our final result aims to be a Jupyter Notebook that, using neural network models we develop in tandem with desired stock market data, would be able to provide portfolio optimization to assist wealth management for users.

The overall challenge is how to define a criterion that shows return on investment (i.e. what counts as a good investment and what is not). There are drawbacks to simply using historical data as a form of measurement for performance, as contextual factors may affect the performance of the particular stock currently.

We split our technical challenges into two categories: predicting individual stock prices, and optimizing portfolios.

Regarding individual stock prices, there are many technical indicators such as closing, opening prices, and volume Kumbure et al. [2022]. There are many other potential factors, including economic data, that can contribute to improved predictive performance.

Regarding optimizing the portfolio, classically we consider mean-variance, but there is also the question of managing profitability vs risk. Park et al. [2024], as well as the consideration of trading fees. Thus, we implemented neural network models dedicated to this task of portfolio optimization.

After constructing the final models, using the sample dataset of stock price for listed companies we performed portfolio optimization with this sample portfolio and compared it against the baseline index of S&P 500 for performance evaluation regarding return on investment.

2 Related work

Kumbure et al. [2022] presented a literature review for the use of machine learning techniques in the field of stock market prediction. Over the period 2000-2019, machine learning applications within this field have mostly revolved around neural networks, support vector machines (SVM), and their variants. Other techniques such as random forest and k-nearest neighbor (KNN) classifiers were also used, albeit less frequently. Recently, deep learning applications for stock market predictions have gained interest, with research conducted on the performance of architectures like convolutional neural networks (CNN) and long short-term memory (LSTM) on stock price prediction. As such, these are methods that we experimented with to find the best-performing architecture.

Ma et al. [2021] and Chen et al. [2021] used deep learning neural networks like multilayer perceptrons (MLP) and CNN (convolutional neural network), as well as other machine learning techniques such as random forest and support vector machines (SVR) to approach portfolio optimization. To be precise, these machine learning techniques were used to predict returns for stocks, which is then used as the basis for classical optimization models such as mean-variance optimization.

3 Methodology

Approaching this problem, we first searched for academic journals that utilize neural networks and other machine-learning techniques for the purpose of stock prediction and portfolio optimization.

Our data pre-processing and neural network approach is derived from two optimization of portfolio papers, both with the possible application of reinforcement learning Wu et al. [2021] Ma et al. [2021]

The portfolio of interest that this paper targets is the Student Foundation Investment Committee portfolio, an endowment portfolio where 100% of funds managed by the committee support UC San Diego students in the form of scholarships

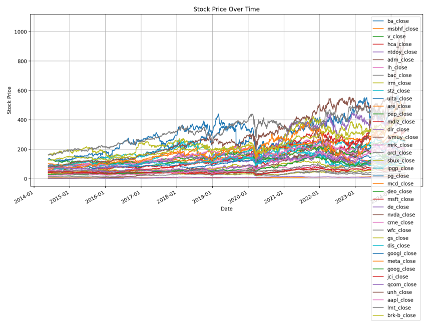


Figure 1: Portfolio stock price valuation over time (in USD)

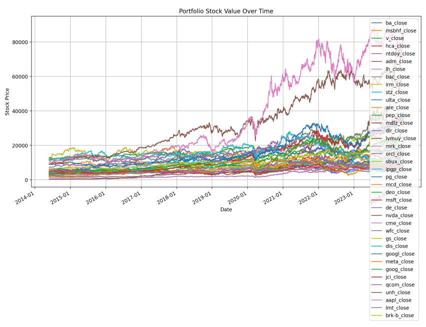


Figure 2: Portfolio net valuation by stock over time (in USD)

We also consult papers on stock portfolio optimization with mean-variance ML techniques (Ma et al. [2021], and we've decided to implement the following procedure for developing a model that can accurately redistribute portfolio weights for best valuation:

3.1 Dataset and Portfolio

The Dataset is obtained from Yahoo Finance, consists primarily of daily information. The dataset collects the following information: Date, Open price, High price, Low price, Close price, Adj Close price and Volume. The period of time chosen was 10 years.

In the Portfolio, the main information provided is the Symbol (Ticker), and Quantity. To simplify the mathematical modeling, it was assumed that the quantities decided by the stock analysts were held constant for the entire dataset. This provided a reasonable rate of return of +150%.

3.2 MVO and MVO-based Neural Network

Mean-variance Optimization (Markowitz [1952]) is a common model used for the task of stock-portfolio optimization. This model seeks to both maximize the return on investment (reward) as well as minimize the risks (volatility), which are represented using the expectation of returns and the covariance between different stock returns. This can be characterized as the following:

$$\text{objective}(\text{weights}) = \sqrt{\text{weights}^T \cdot \text{cov\_matrix} \cdot \text{weights}} - \sum(\text{mean\_returns} \cdot \text{weights})$$

$$\sum \text{weights} = 1$$

Building upon this general model, we add a risk averse factor λ to address the preference of the model for either reward or risk (Chen et al. [2021]).

$$\text{objective}(\text{weights}) = \text{risk} \cdot \sqrt{\text{weights}^T \cdot \text{cov\_matrix} \cdot \text{weights}} - (1 - \text{risk}) \cdot \sum(\text{mean\_returns} \cdot \text{weights})$$

MVO-based Neural Network:

We train the MVO-based neural network to perform Mean-Variance Optimization using a fully connected (FC) network with ReLU activation functions, rather than performing mean, variance and using an optimizer. The MVO-based neural network removes the computational complexities of the optimizer by being pre-trained, in order to perform calculations at only O(n).

The loss function is given by the following:

$$\text{Loss} = \left(\sum_{i=1}^{n} (\hat{y}_i - y_i)^2 \right) \cdot \frac{\text{initial\_value}}{\sum_i \text{portfolio}_i \cdot \hat{y}_i}$$

where:

- portfolio represents the values of the portfolio at
- $\hat{y}_i$ are the predicted weight values.
- y_i are the MVO-derived weight values.
- i is the index of the day
- n is the initial i + 60

The Mean Squared Error is a utilized as the loss function for our regression tasks, measuring the average squared difference between the predicted values ($\hat{y}$) and the ideal MVO-derived values (y). This part of the loss function promotes learning of the ideal values early on, when we are close to the initial portfolio value.

At the same time, the MSE is then scaled by the initial value and normalized by the predicted total value to ensure proportional evaluation, and to promote an increased portfolio value.

3.3 ML-based Stock Price Prediction with Post-Prediction Optimization Using MVO

This is a two-stage method inspired by Wu et al. [2021]. It first predicts the next-day closing prices of an individual stock using the last 60 days of history prices, and then applies a modified version of MVO upon the predictions for portfolio formation.

For stock prediction, 6 features including closing prices, adjusted closing prices, opening prices, highs, lows, and volumes are used as input. The model we tried out includes Support Vector Regressor (SVR), Random Forest (RF), Deep Multi-layer Perceptron (DMLP), and LSTM.

Given the prediction results, the MVO optimization could be written as follows:

$$\min \sum_{i,j=1}^{n} x_i x_j \sigma_{ij} - \sum_{i=1}^{n} x_i \hat{r}_i - \sum_{i=1}^{n} x_i \bar{\epsilon}_i$$

Subject to:

$$\sum_{i=1}^{n} x_i = 1$$

$$0 \leq x_i \leq 1 \quad i = 1, 2, \ldots, n$$

where:

- x_i represents the proportion of asset i in the portfolio,
- n is the number of assets in the portfolio,
- σ_{ij} is the covariance between asset i and asset j,
- $\hat{r}_i$ is the predicted return of asset i,
- $\bar{\epsilon}_i$ is the average predictive error of asset i over the sample period.

3.4 STM Neural Network with Embedded Optimizer

We also tried using a Long Short-Term Memory (LSTM) neural network Vennerød et al. [2021] combined with an MVO embedded optimizer for optimizing stock portfolio weights. LSTM networks are particularly suitable for sequence prediction problems. It's really useful for stock price forecasting due to its ability to remember long-term dependencies while forgetting irrelevant information.

3.4.1 LSTM Neural Network Architecture

LSTMs are a type of recurrent neural network (RNN) designed to overcome the limitations of traditional RNNs, such as the vanishing gradient problem. This makes them great for tasks involving time series forecasting.

The architecture of an LSTM cell includes several components:

1. **Forget Gate** (f_t): Decides what information from the previous cell state should be forgotten.

2. **Input Gate** (i_t): Determines which new information should be added to the cell state.

3. **Cell State** (C_t): Stores the long-term memory.

4. **Output Gate** (o_t): Decides the output based on the cell state and the input.

The mathematical operations within an LSTM cell are as follows:

1. **Forget Gate**:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

where σ is the sigmoid function, W_f are the weights, h_{t-1} is the previous hidden state, x_t is the input, and b_f is the bias.

2. **Input Gate**:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

where $\tanh$ is the hyperbolic tangent function, and W_i, W_C, b_i, b_C are the weights and biases for the input gate.

3. **Cell State Update**:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$$

4. **Output Gate**:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t \cdot \tanh(C_t)$$

3.4.2 Why LSTMs?

LSTMs are fantastic for stock predictions due to their ability to capture temporal dependencies and patterns in sequential data. Stock prices are influenced by various factors over time, and LSTMs can effectively model these relationships, leading to more accurate predictions.

Key advantages of LSTMs for stock predictions:

- **Memory Mechanism**: LSTMs can retain important information over long periods, which is suitable for predicting future stock prices based on historical data.

- **Handling Non-linearity**: Stock prices often exhibit non-linear patterns, which LSTMs can model effectively.

- **Robustness to Noise**: LSTMs can filter out irrelevant information, making them robust to noisy data, common in financial markets.

3.4.3 Training the LSTM Model

The LSTM model is trained using the following steps:

1. **Data Preprocessing**: Normalize the stock price data and create input sequences.

2. **Model Training**: Train the LSTM model using the training set with the Mean Squared Error (MSE) loss function and Adam optimizer.

3. **Hyperparameter Tuning**: Optimize the number of layers, the learning rate, and the batch size using the validation set.

4. **Model Evaluation**: Evaluate the model's performance on the testing set.

3.4.4 Embedded Optimizer for Portfolio Optimization

After training the LSTM model, we use its predictions to perform portfolio optimization. We integrate an optimizer directly into the neural network to adjust the portfolio weights dynamically based on the predicted stock prices.

The embedded optimizer adjusts the portfolio weights in real-time, ensuring optimal allocation based on the latest predictions. This approach combines the predictive power of LSTMs with the efficiency of real-time optimization, leading to improved portfolio performance.

After implementing all approaches, we compare the results using our sample dataset as portfolio and adopt the best approach based on the final return on investment.

4 Experiments

The experiment was conducted by validating in the three phases prescribed. We hypothesize that the architecture of neural network with embedded optimizer would perform the best, as it is a concatenation and re-development of the previous two approaches.

4.1 MVO

In the initial phase, MVO was performed across the overall range of specified time (10 years). In the images below, the "Portfolio" refers to the optimized portfolio based on MVO. In this case, we compare against S&P 500.

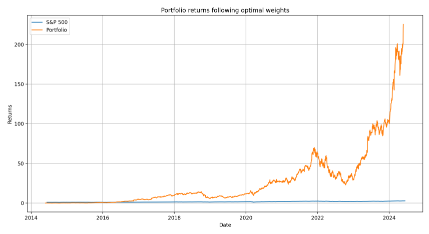


Figure 3: MVO vs S&P 500

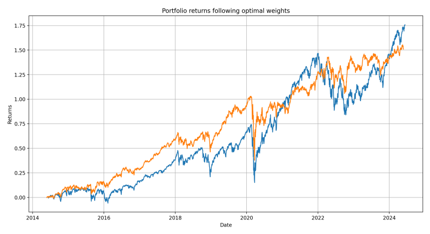


Figure 4: MVO w/ risk vs S&P 500

Subsequently, MVO was performed across sliding window of 60 days over the 10 years, and the optimal weight distribution was chosen. The weight corresponds to the number of stocks that can be purchased normalized to the initial time point.

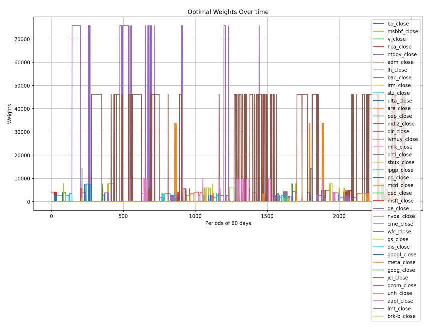


Figure 5: MVO optimal weights for 60-day sliding-window

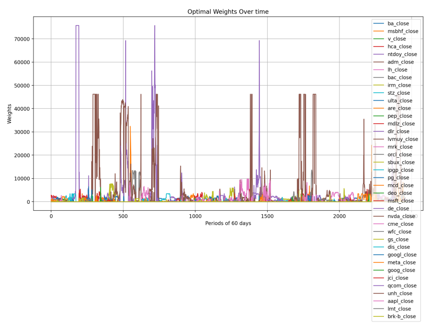


Figure 6: MVO w/ risk optimal weights for 60-day sliding-window

However, the weight distribution of MVO prioritized a singular stock that performs well during the window. Utilizing MVO with risk, the stock weight is more distributed, especially during times with greater uncertainty.

4.2 MVO-based Neural Network

After performing Training in a 6-2-2 split in terms of the 10 years in the dataset, for Training (6 years), Validation (2 years), and Testing (2 years, this is what we are plotting).

The Neural network chosen was a 2-layer Fully connected layer each with a ReLU activation layer. This was chosen as a simple model to evaluate the potential of this method. The top 10 weights (by contribution to portfolio valuation) were plotted.

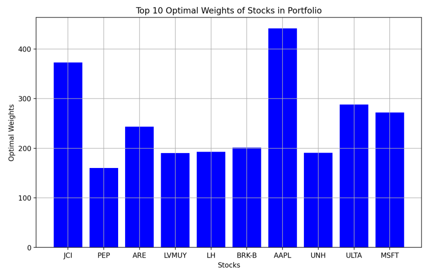


Figure 7: NN Weights without Risk

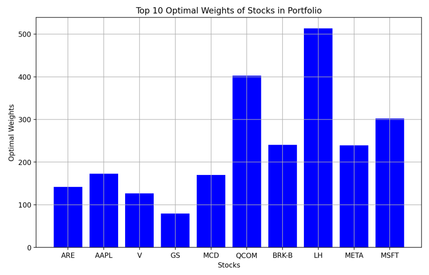


Figure 8: NN Weights with Risk

The first thing to note is that the error converges rapidly within the first few epochs. Adjusting the shape and size of network did not seem to affect the convergence significantly.

Furthermore, the distribution of weights do not significantly differ between the neural network trained with risk and without risk. The corresponding final value of the portfolio is also consistently higher than the initial.

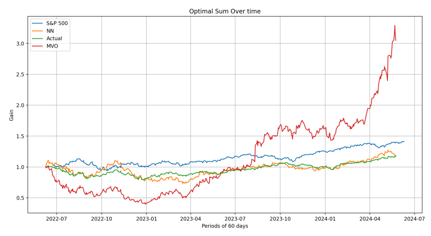


Figure 9: NN Testing

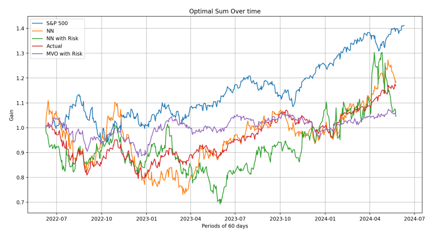


Figure 10: NN Testing

Therefore, the MVO seems to suggest putting all eggs in a single basket, such as Nvidia. However, realistically, this is not ideal as it will make the portfolio itself extremely vulnerable to potential policy changes or global phenomenons that impact a single industry. An additional drawback to MVO is the large computational complexity associated with calculating weights for large datasets.

With and without risk, the MVO-based training performs similarly to the portfolio managed by the analysts. It does not appear to provide significant improvement from the baseline analyst portfolio, but both performed worse than the benchmark S&P 500 within the last 2 years of our testing set. It also appears that the neural-network based portfolio does not improve the variance of the portfolio value. The prices are able to shift more up and down than the portfolio chosen by the analysts.

Since the trained neural network achieves performance similar to the baseline analyst portfolio, it is likely that the mechanism is similar. Seen in Figures 3 to 6, it also shows that the regular MVO performs just as well, and consistently higher. In terms of risk-mitigation, the analysts still perform better. In terms of maximizing profit, advanced methods such as neural networks may not yield better results. Therefore, effort was concentrated in the price prediction, and utilizing MVO as the optimizer.

4.3 Using LSTMs to Predict Highest Return on Investment per Stock

We integrated multiple LSTM neural networks to predict a return on investment if a user were to buy and sell stocks on certain days. We trained one LSTM network on every stock. For each day, the model would determine whether it should buy or sell on that particular day based on the opening and closing stock prices each day. The results on return on investment is shown.

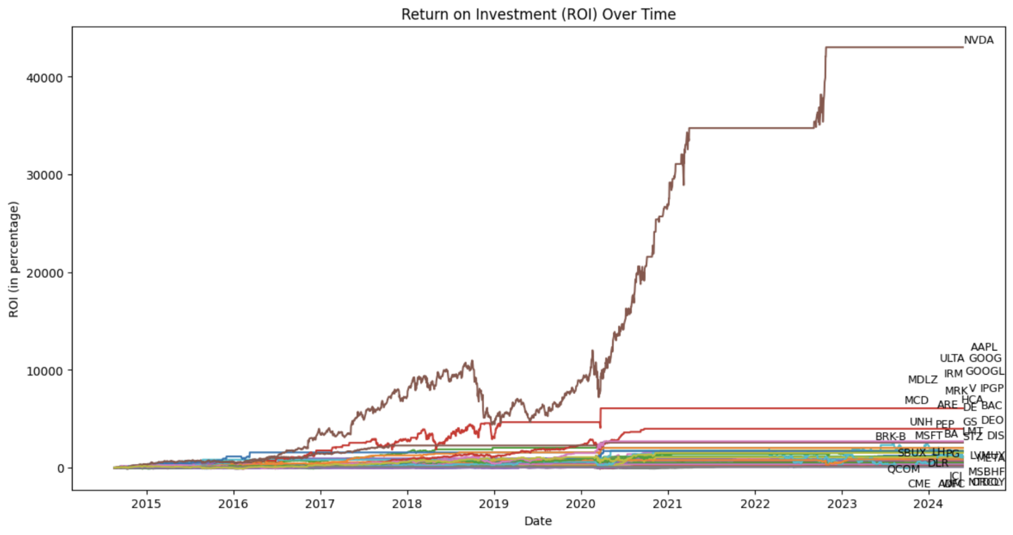


Figure 11: Stock Predictions of Different Models on Multiple Stocks

As we can see, over the last 10 years, NVIDIA has given us the most return on investment, showing that NVIDIA does have a significant impact on our model results, since it will choose to invest in NVIDIA for maximum profit.

4.4 ML-based Stock Price Prediction with Post-Prediction Optimization Using MVO

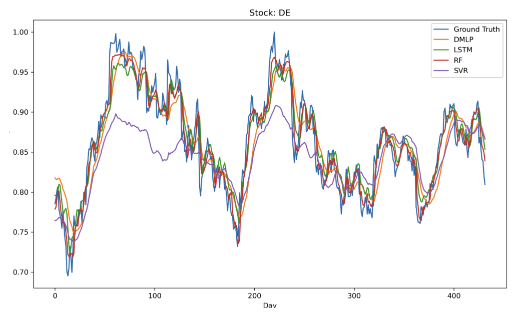
The dataset is split into a 6-2-2 train-validation-test split. Individual prediction models are trained for each stock on the training set and we use the average mean squared error (MSE) on the validation set for all stocks to guide parameter tuning. Then the price predictions on the test set are used in MVO calculation to acquire the optimal weight of assets over time on the test set.

First, we show the prediction results of different models, shown in the table below. RF models achieved the best overall performance followed by LSTM and DMLP, while SVR models yield the worst results. The superiority of RF could be attributed to DL models' vulnerability to overfitting on this relatively small data.

Table 1: Performance of different Models on the Test Set

Model	**MSE**
SVR	2.2×10^{-3}
RF	2.8×10^{-4}
LSTM	4.6×10^{-4}
DMLP	5.3×10^{-4}

The figure below demonstrates the prediction of closing prices on two individual stocks. We could also tell that RF, LSTM, and DMLP did well in following the changing trend but RF did the best. SVR's prediction is slightly off track, especially at peaks and valleys.



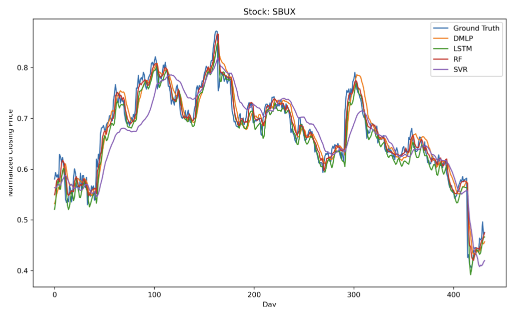


Figure 12: Stock Prediction of Different Models on Two Sample Stocks

Finally, the MVO result is shown in the table and figure below. Overall RF+MVO achieved dominating performance (net value of 4.80) over other models. DMLP+MVO also achieves better performance than S&P 500 (net value of 2.06). LSTM+MVO and SVR+MVO even lead to a decrease in net value, which we suspect is due to SVR's prediction inaccuracy and LSTM's tendency to underestimate return.

Table 2: Net Value of each model with MVO

Model	**MSE**
SVR+MVO	0.56
RF	**4.80**
LSTM+MVO	0.69
DMLP+MVO	2.06
S&P 500	1.17

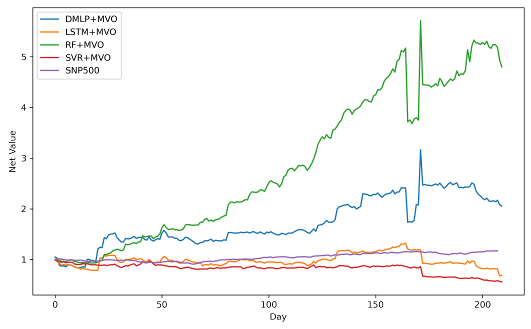


Figure 13: Net Values of Different MVO Models

5 Conclusion

Of the three approaches to stock portfolio optimization we have evaluated, the neural network with embedded optimizer performed the best. Whilst it doesn't have the outright best return on investment compared to standard MVO, the prediction speed is orders of magnitude faster as the time complexity is almost $O(n)$ against $O(n^2)$.

Compared to the S&P 500, while the MVO-based neural networks was unable to beat its growth, utilizing neural networks (e.g. MLP) with a separate optimization resulted in noticeable gains compared to the index.

Compared to the actual portfolio held by the school committee, the MVO-based neural networks performed similarly, while both were outperformed considerably by the basic MVO. This gap is mainly due to the presence of an outlier company (Nvidia) that performed exceedingly well compared to all other stocks. Eliminate this company from the portfolio, then MVO-based neural networks, S&P 500, and the actual portfolio will converge in performance.

Through the project, we aim to shed light on optimal neural network structures and portfolio optimization algorithms tailored for portfolio management tasks. For practical applications of the project, it will be able to serve as a handy complementary tool for portfolio management in the real world. Working in tandem with dedicated wealth management services and applications, this project will provide traders with more information about their portfolios, thus helping them make more educated decisions about asset management.

References

W. Chen, H. Zhang, M. K. Mehlawat, and L. Jia. Mean–variance portfolio optimization using machine learning-based stock price prediction. *Applied Soft Computing*, 100:106943, 2021. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2020.106943. URL `https://www.sciencedirect.com/science/article/pii/S1568494620308814`.

M. M. Kumbure, C. Lohrmann, P. Luukka, and J. Porras. Machine learning techniques and data for stock market forecasting: A literature review. *Expert Systems with Applications*, 197:116659, 2022. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2022.116659. URL `https://www.sciencedirect.com/science/article/pii/S0957417422001452`.

Y. Ma, R. Han, and W. Wang. Portfolio optimization with return prediction using deep learning and machine learning. *Expert Systems with Applications*, 165:113973, 2021. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2020.113973. URL `https://www.sciencedirect.com/science/article/pii/S0957417420307521`.

H. Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952. ISSN 00221082, 15406261. URL `http://www.jstor.org/stable/2975974`.

K. Park, H.-G. Jung, T.-S. Eom, and S.-W. Lee. Uncertainty-aware portfolio management with risk-sensitive multiagent network. *IEEE Transactions on Neural Networks and Learning Systems*, 35(1):362–375, 2024. doi: 10.1109/TNNLS.2022.3174642.

C. B. Vennerød, A. Kjærran, and E. S. Bugge. Long short-term memory rnn, 2021.

M.-E. Wu, J.-H. Syu, J. C.-W. Lin, and J.-M. Ho. Portfolio management system in equity market neutral using reinforcement learning. *Journal of Computational Science*, 52:101240, 2021. doi: 10.1016/j.jocs.2021.101240.