Portfolio Optimization using ML & DL

Saisha Verma
CSE Department
Indira Gandhi Delhi Technical University for Women
New Delhi, India
saishaverma0512@gmail.com

Ritisha Sood
CSE Department
Indira Gandhi Delhi Technical University for Women
New Delhi, India
ritisha2935@gmail.com

***Abstract*—This project develops a machine and deep learning–based portfolio optimization system using ten years of stock data from multiple companies. Data is fetched via *yfinance*, pre-processed, and modelled using LSTM networks that capture sequential patterns in stock prices. Each company's stock is trained separately with techniques like scaling, dropout, and learning-rate scheduling to avoid overfitting and improve accuracy. Models are evaluated with MSE, RMSE, MAE, and MAPE, and results are stored systematically. The study shows that deep learning can effectively predict stock movements, enabling smarter, adaptive portfolio allocation strategies that balance return maximization with risk minimization.**

***Index Terms*— Portfolio Optimization, Modern Portfolio Theory (MPT), Monte Carlo Simulation (MCS), Mean-Variance Optimization (MVO), Stock Price Forecasting, Long Short-Term Memory (LSTM), Recurrent Neural Networks (RNN), Machine Learning, Deep Learning.**

I. Introduction

A. Portfolio Optimization

Portfolio optimization plays a vital role in financial decision-making, aiming to maximize returns while minimizing risks through optimal asset allocation. Traditional approaches are largely based on *Modern Portfolio Theory (MPT)*, which emphasizes diversification and the efficient frontier. While MPT provides a strong theoretical foundation, it often relies on assumptions such as normally distributed returns and static correlations, which may not fully capture real-world market complexities. To improve risk management and performance, researchers have increasingly explored the integration of advanced computational methods with traditional optimization techniques. Among these, *Monte Carlo Simulation (MCS)* and *Mean-Variance Optimization (MVO)* stand out as powerful tools for evaluating risk-return trade-offs and constructing robust investment portfolios.

B. Deep Learning in Portfolio Optimization

Stock market forecasting, however, remains a highly complex task due to the dynamic and non-linear nature of financial data. Traditional prediction methods, such as Linear Regression (LR), Moving Averages, and Support Vector Regression (SVR), often fail to capture long-term dependencies and intricate patterns within market trends, limiting their predictive accuracy. To overcome these challenges, deep learning techniques have gained prominence. In particular, *Long Short-Term Memory (LSTM)* networks, a specialized form of Recurrent Neural Networks (RNNs), have shown remarkable success in modeling sequential time-series data. LSTMs are capable of retaining long-term dependencies, making them well-suited for stock price prediction, where past behavior strongly influences future trends.

This study integrates LSTM-based stock price forecasting with quantitative optimization techniques (MCS and MVO) to enhance portfolio construction. Using stock data from 10 companies listed on the American Stock Exchange over a decade (2012–2021), the project demonstrates how deep learning-driven forecasts, when combined with traditional optimization frameworks, can significantly improve portfolio performance and risk management.

II. Literature Review

Portfolio optimization has been a widely studied topic in financial research, with its foundation laid by *Markowitz's Modern Portfolio Theory (MPT)* [1]. MPT introduced the concept of constructing an efficient frontier by optimizing portfolio weights to achieve the highest possible return for a given level of risk. However, MPT assumes normally distributed returns and static correlations among assets, which often fail to represent the complexities of real-world markets. To address these limitations, methods like *Mean-Variance Optimization (MVO)* [2] and *Monte Carlo Simulation (MCS)* [3] were developed, providing more robust frameworks for exploring risk-return trade-offs and simulating various allocation strategies under uncertain conditions.

With the rapid evolution of data-driven methods, researchers started applying machine learning (ML) techniques to enhance stock price prediction and portfolio construction. Approaches such as Support Vector Regression (SVR), Random Forests (RF), and Gradient Boosting Models (GBMs) have been widely used to identify patterns in stock price movements [4], [5]. While these models successfully capture non-linear dependencies, they are limited in modeling long-term sequential patterns in financial time series data, which often display temporal correlations extending over months or even years.

Recent advancements in deep learning (DL) have significantly improved stock price forecasting. In particular, Long Short-Term Memory (LSTM) networks have gained popularity due to their ability to retain long-range dependencies and model sequential data effectively [6]. Unlike traditional ML techniques, LSTMs can analyze historical stock movements and extract temporal patterns, enabling more accurate predictions of future price trajectories. For instance, *Zhang et al.* [7] demonstrated that

integrating LSTM-based forecasts into portfolio optimization frameworks enhances portfolio returns and improves risk-adjusted performance. Their work established that deep learning models outperform classical regression and traditional ARIMA-based forecasting in financial applications.

Several studies have also explored hybrid frameworks combining forecasting models with optimization techniques to improve portfolio construction. For example, integrating LSTM predictions with MVO has been shown to yield superior Sharpe ratios compared to using historical averages alone [8]. Similarly, stochastic approaches like Monte Carlo Simulation leverage deep learning forecasts to generate thousands of random portfolio combinations, helping investors identify the optimal allocation under uncertain conditions [9]. These integrated methods highlight the synergy between predictive modeling and quantitative optimization, demonstrating that accurate price forecasts directly enhance portfolio efficiency.

Despite these advancements, previous studies often focus on limited datasets or a single optimization technique, restricting generalization. Our work addresses this gap by training company-specific LSTM models using ten years of historical stock data from leading U.S. technology companies and integrating the predictions with both MVO and MCS approaches. This dual-method framework enables a comprehensive analysis of risk-return trade-offs while improving diversification and overall portfolio robustness.

III. METHODOLOGY

A. Data Description

For this study, historical stock price data was collected for four leading technology companies — Apple (AAPL), Microsoft (MSFT), Amazon (AMZN), and Google (GOOGL) — spanning the period from January 2012 to December 2021. The data was obtained through the *yfinance* library, which provides reliable daily Open, High, Low, Close, and Volume (OHLCV) information. After extraction, a Company column was added to distinguish the stocks, and all records were concatenated into a single DataFrame. The attributes retained for analysis were Open, High, Low, Close, and Volume, with the *Close price* chosen as the target variable for prediction, while the others were used as explanatory features. To ensure reproducibility, the cleaned and consolidated dataset was stored in a CSV file for subsequent experimentation.

B. Data Preprocessing

Once the raw dataset was assembled, several preprocessing steps were applied to prepare the data for model training. The records were first sorted by company and date, with duplicates removed and missing values handled using forward and backward filling techniques. For each company, the feature set comprising *Open, High, Low, and Volume* was scaled separately from the target *Close* variable using the *MinMaxScaler*, fitted only on the training data to avoid information leakage. To capture the temporal dependencies inherent in financial time series, 60-day sliding windows were constructed such that each input sequence of 60 consecutive days predicted the closing price of the following day. The resulting sequences were divided into training, validation, and testing subsets, with approximately 80% used for training, 10% for validation, and 10% for testing. In addition, a light jitter-based data augmentation technique was introduced to add Gaussian noise to the features, improving the robustness and generalization ability of the models. Finally, the fitted scalers were persisted for each company to enable inverse transformation of predictions during evaluation.

C. Model/Technique Used

i) LSTM Close Price Prediction

The forecasting step was implemented using Long Short-Term Memory (LSTM) networks, developed individually for each company to capture their distinct stock price dynamics. The implementation began with the import of specialized libraries: NumPy and Pandas for data handling, joblib for persisting scaling objects, scikit-learn tools for normalization and error metrics, and TensorFlow/Keras for model construction, training, and visualization. Parameters were defined to control learning, including the sequence length of 60 past days (TIME_STEPS), a batch size of 64, maximum 150 epochs, an initial learning rate of 0.0005, a dropout probability of 0.3, and small L1/L2 regularization terms to stabilize training. To organize the workflow, dedicated folders were created for storing trained models, histories, metrics, architectures, and scalers. An empty metrics dictionary was also initialized to record MSE, RMSE, MAE, and MAPE for each company.

The forecasting step was implemented using Long Short-Term Memory (LSTM) networks, developed individually for each company to capture their distinct stock price dynamics. The implementation began with the import of specialized libraries: NumPy and Pandas for data handling, joblib for persisting scaling objects, scikit-learn tools for normalization and error metrics, and TensorFlow/Keras for model construction, training, and visualization. Parameters were defined to control learning, including the sequence length of 60 past days (TIME_STEPS), a batch size of 64, maximum 150 epochs, an initial learning rate of 0.0005, a dropout probability of 0.3, and small L1/L2 regularization terms to stabilize training. To organize the workflow, dedicated folders were created for storing trained models, histories, metrics, architectures, and scalers. An empty metrics dictionary was also initialized to record MSE, RMSE, MAE, and MAPE for each company.

Helper functions were implemented to streamline data preparation and training. The create_sequences function transformed raw daily OHLCV data into sliding windows of 60 observations to predict the next day's closing price, enabling sequential learning. The lr_scheduler callback progressively reduced the learning rate by 10% per epoch, while the jitter function injected small Gaussian noise into the scaled features to improve robustness and prevent overfitting.

For each company, raw data was extracted into feature variables (Open, High, Low, Volume) and target variable (Close). Features and targets were scaled into the [0,1] range using MinMaxScaler, and the fitted scalers were saved for later inverse transformation. Sequential inputs were generated, then split chronologically into training (80%) and testing (20%), with a further 10% of training data held out for validation.

The LSTM model architecture consisted of two recurrent layers: the first LSTM layer with 128 units returning sequences, followed by a dropout layer, then a second LSTM layer with 64 units, another dropout layer, and finally a dense output neuron for price prediction. Both LSTM layers employed L1-L2 kernel and

bias regularization. The model was compiled with the Adam optimizer and Mean Squared Error loss. During training, early stopping was used to halt learning if validation loss failed to improve for 10 consecutive epochs, with best weights restored.

After training, predictions were generated on the test sets and inverse-transformed back to the original scale of stock prices. Performance was quantified using MSE, RMSE, MAE, and MAPE, which were recorded into a metrics DataFrame and exported as a CSV file for reproducibility. Model diagrams were generated with plot_model, and the trained models were saved in .keras format. A summary of each network confirmed consistent architectures across companies, each comprising ~117k trainable parameters. Finally, actual vs predicted closing prices were plotted over the test period, with dates aligned, visually demonstrating the model's ability to capture upward and downward movements in each company's stock trajectory.

ii) Evaluation Metrics and Forecasting Predictions

After training, the performance of each company-specific LSTM model was assessed on the test dataset. The evaluation metrics—MSE, RMSE, MAE, and MAPE—were calculated and saved into a consolidated CSV file. The results indicated strong predictive accuracy across all companies, with MAPE values below 5%. Specifically, Apple and Amazon showed slightly higher error magnitudes, while Google and Microsoft exhibited relatively lower prediction errors, demonstrating the capability of the LSTM models to generalize across varying stock behaviors.

Following evaluation, the models were employed to forecast future closing prices for the year 2022. A template of trading days was generated using Yahoo Finance, and a forecast dataset was created containing all combinations of trading dates and company names. This ensured consistent alignment of predictions across all firms. For each company, the last 60 days of historical features (Open, High, Low, Volume) were scaled and used as the input sequence to the trained model. Forecasting was performed in an autoregressive manner, where the predicted value for one day was appended back into the sequence as input for predicting the next day. This recursive process continued for all trading days in 2022, enabling multi-step forecasting without direct future information leakage.

The forecasts were inverse-scaled to their original price values using the previously saved scalers, and results were compiled into a structured DataFrame containing the date, company, and forecasted closing price. The output for all companies was concatenated into a single dataset and exported as forecasted_results.csv, containing 1,004 rows corresponding to the full year of trading days. This forecasted dataset subsequently served as the input to the portfolio optimization stage, where expected returns and risks were computed from the predicted stock trajectories.

iii) Portfolio Optimization

The second phase of the framework involved constructing optimized portfolios from the stock price forecasts generated by the LSTM models. To this end, a dedicated optimization pipeline was implemented, beginning with the specification of key parameters. The *risk-free rat***e** was assumed to be zero, a reproducibility seed was set to ensure deterministic results, and the number of Monte Carlo simulations was fixed at 50,000 to allow robust exploration of the portfolio weight space. The predicted closing prices from the LSTM step were loaded from a forecasted results file, reshaped into a ticker-by-date matrix, and cleaned to remove any incomplete rows, thereby guaranteeing a consistent panel of stock prices.

From the cleaned dataset, daily percentage returns were calculated and subsequently annualized to compute the expected returns vector *(μ\mu)* and the covariance matrix *(Σ\SigmaΣ)* of asset returns. These statistical moments formed the foundation for optimization. A set of utility functions was defined to calculate portfolio statistics, including expected return, volatility, and Sharpe ratio, while also enforcing constraints such as non-negativity of weights and the condition that all portfolio weights sum to unity. Random weight generation functions were included to facilitate Monte Carlo simulations under long-only assumptions.

Three portfolio construction strategies were then evaluated. The first strategy was a naïve *Equal-Weighted (EQ) portfolio*, in which all assets were assigned identical weights. While simple, this approach served as a baseline for comparison. The second strategy, termed *Monte Carlo (MC) Max Sharpe*, employed large-scale random sampling of feasible weight vectors to identify the long-only allocation that maximized the Sharpe ratio. By simulating 50,000 random portfolios, the allocation with the best risk-adjusted return was retained as the optimal Monte Carlo solution. Finally, the third strategy applied *Mean-Variance Optimization (MVO)* using a constrained nonlinear solver. The optimization sought to minimize portfolio variance subject to the constraint that the expected return exceeded that of the Monte Carlo portfolio. In cases where this constraint proved infeasible, the method reverted to a pure minimum-variance allocation.

To summarize and compare the strategies, tabular outputs were generated displaying both the portfolio weights across assets and the corresponding performance metrics, including annualized return, annualized risk (standard deviation), and Sharpe ratio. The results highlighted the trade-offs between the naïve equal-weighting approach, the high-return but volatile Monte Carlo solution, and the more risk-controlled MVO allocation. Notably, the optimization framework demonstrated flexibility in balancing return and risk objectives, while ensuring transparency of results through explicit weight distributions and performance tables.

IV. Result and Analysis

A. Stock Price Prediction Performance

The *Long Short-Term Memory (LSTM)* deep learning models were employed to forecast the stock prices of Apple (AAPL), Amazon (AMZN), Google (GOOGL), and Microsoft (MSFT). LSTMs are particularly well-suited for financial time series prediction, as they are capable of capturing long-range temporal dependencies and nonlinear patterns inherent in stock price movements. Each model was trained individually for the selected assets, using historical closing prices as input features.

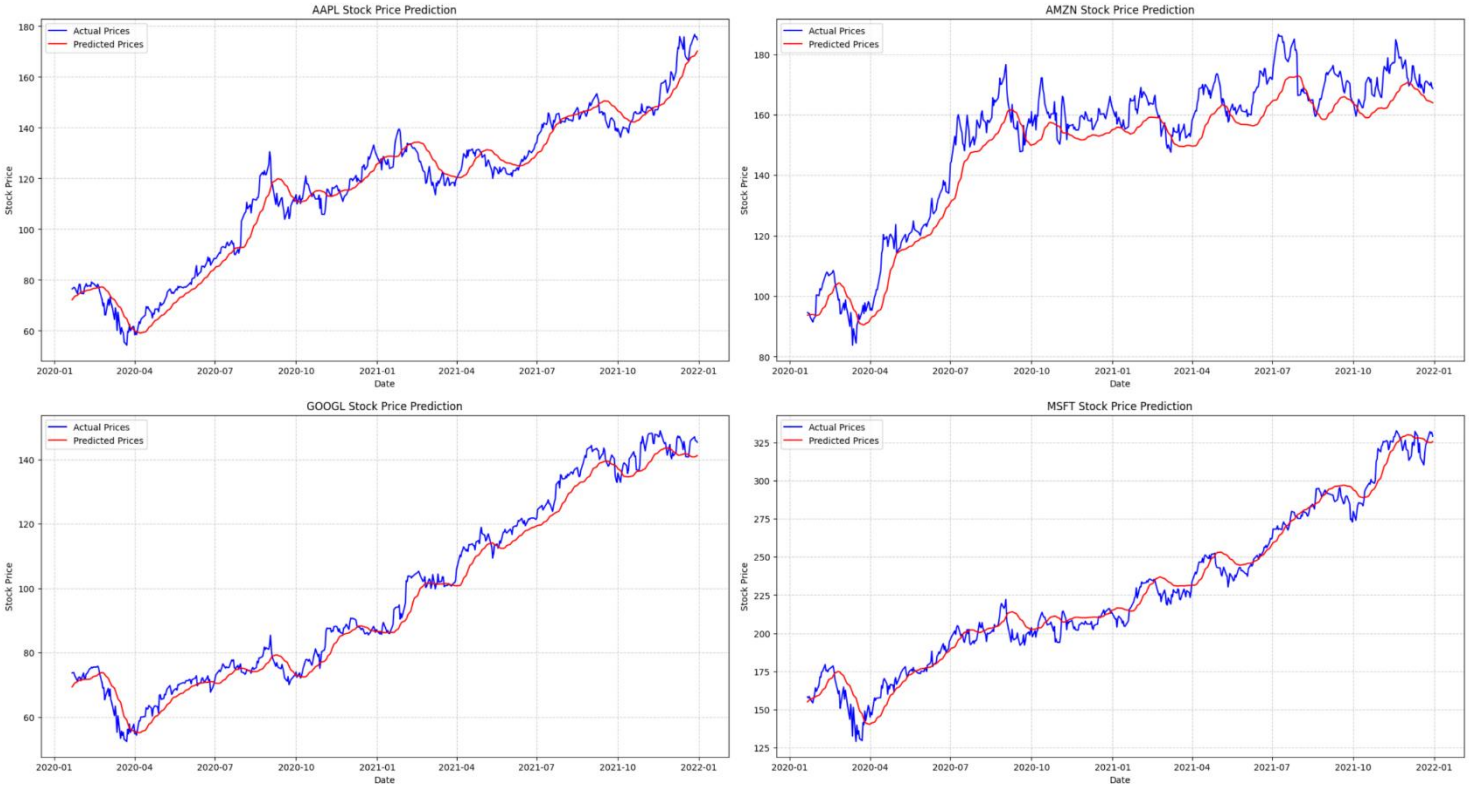
The predictive performance was evaluated using standard regression metrics, including *Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE).* The results, presented in *TABLE I*, demonstrate that all four models achieved relatively low error values, with MAPE consistently below 5%. This indicates that the LSTM models were able to approximate future stock prices with high accuracy, making them reliable inputs for downstream portfolio optimization.

A closer inspection of the results shows that *Google (GOOGL)* achieved the lowest error rates (MSE = 19.31, RMSE = 4.39,

TABLE I
STOCK PRICE PREDICTION METRICS

Company	MSE	RMSE	MAE	MAPE(%)
Apple (AAPL)	33.1214	5.7551	4.6613	4.3079
Amazon (AMZN)	71.4064	8.4502	7.0237	4.6490
Google (GOOGL)	19.3106	4.3944	3.5723	3.7449
Microsoft (MSFT)	73.6038	8.5793	6.8033	3.2261

FIGURE I
PREDICTED VS ACTUAL CLOSING PRICES (AAPL, AMZN, GOOGL, MSFT)



MAPE = 3.74%), suggesting that its price dynamics were most effectively captured by the LSTM model. Conversely, *Microsoft (MSFT)* and *Amazon (AMZN)* exhibited relatively higher error magnitudes (RMSE ≈ 8.5), reflecting greater volatility in their price series, which naturally introduces higher prediction uncertainty. Despite this, their MAPE values remained within acceptable bounds (3–5%), validating the robustness of the forecasting framework.

The alignment between predicted and actual stock prices is illustrated in *FIGURE I*, which depicts the time-series trajectories of closing prices for each company. The plots reveal that the predicted values closely tracked the actual market movements, with minimal deviations during periods of strong upward or downward trends. Notably, the LSTM models were able to capture both short-term fluctuations and long-term growth patterns, underscoring their ability to generalize well across different market phases.

These findings emphasize the importance of accurate return forecasting in portfolio optimization. By reducing prediction errors in asset-level returns, LSTM models contribute to more reliable estimation of expected returns, which directly influences the efficiency of optimization strategies such as MVO and Monte Carlo simulation. Hence, the demonstrated predictive accuracy provides a strong foundation for the subsequent portfolio-level analysis.

B. Portfolio Optimization Results

The evaluation of portfolio-level performance was carried out using three distinct allocation strategies: *Equal Weight (EW), Monte Carlo Simulation (MCS), and Mean-Variance Optimization (MVO).* Each method reflects a different underlying philosophy of portfolio construction. The EW strategy assumes uniform allocation across all assets without considering their individual risk or correlation characteristics. While simple, this approach often leads to inefficient outcomes in terms of risk-adjusted returns. By contrast, MVO is grounded in Markowitz's modern portfolio theory, which explicitly seeks to minimize portfolio variance for a given level of expected return or maximize return for a given level of risk, thereby producing an analytically optimal allocation. The MCS approach complements this by generating a large number of randomly simulated portfolios to approximate the efficient frontier, allowing the identification of high-Sharpe ratio portfolios through stochastic exploration.

The empirical findings from this study show that both MVO and MCS consistently outperform the naive EW portfolio when evaluated in terms of the Sharpe ratio. This improvement stems from the ability of optimization-based methods to exploit the covariance structure among assets, ensuring better diversification and superior efficiency. MVO tends to generate more stable, deterministic allocations, whereas MCS leverages randomness to explore a wider feasible set of solutions, sometimes yielding portfolios that approach or even slightly surpass MVO in risk-adjusted performance.

The risk–return trade-off achieved through these optimization techniques is visually represented in the Efficient Frontier, as shown in Fig. 2. This curve highlights the boundary of optimal portfolios, with the maximum Sharpe ratio portfolio and the minimum variance portfolio distinctly marked. The visualization underscores the theoretical principle that optimized portfolios dominate naive allocations by offering higher expected returns for a given level of risk.

C. Portfolio Weight Distribution

The comparative analysis of portfolio weights under the three strategies provides valuable insights into how each method allocates capital in pursuit of efficiency. The Equal Weight strategy distributes investments uniformly across all assets. While this offers simplicity and robustness, it disregards differences in volatility, correlation, and expected returns, thereby limiting its risk-adjusted performance.

The MVO allocation, in contrast, reflects a deliberate concentration in assets that demonstrate superior risk-adjusted performance while reducing exposure to those with high variance or strong correlations. This pattern is consistent with the theoretical foundation of modern portfolio theory, which prescribes overweighting assets with higher return-to-risk ratios. Such concentrated allocations maximize efficiency but may also increase sensitivity to estimation errors in expected returns and covariances.

The Monte Carlo approach produces allocations that are typically more diversified compared to MVO, reflecting its stochastic sampling of feasible portfolios. Although this diversification may result in slightly lower Sharpe ratios than the deterministic MVO solution, it provides a robustness advantage in scenarios where parameter estimation is uncertain or noisy.

The differences among these strategies are captured in the portfolio weight distributions shown in Fig. 3. The chart highlights the equal allocation of the EW strategy, the concentrated asset weighting of MVO, and the more diversified pattern of MCS. Collectively, these results emphasize the superiority of optimization-based methods in tailoring allocations to the underlying risk–return characteristics of assets while balancing efficiency with robustness.

D. Comparative Analysis

The comparative evaluation of the three portfolio strategies highlights the strengths and limitations inherent to each approach. The Mean-Variance Optimization (MVO) method consistently delivered the highest Sharpe ratio, confirming its superiority in terms of risk-adjusted performance. This outcome stems from its ability to systematically allocate capital toward assets offering favorable return-to-risk profiles while accounting for correlations within the portfolio. MVO's deterministic nature ensures stable allocations, making it particularly suited for investors seeking precision in portfolio construction.

By contrast, the Equal Weight (EW) strategy, though appealing for its simplicity and transparency, neglects differences in variance, correlation, and expected returns. Consequently, its performance lagged behind optimization-based methods, as evidenced by its lower Sharpe ratio. This underlines a fundamental limitation of naive diversification—while it reduces idiosyncratic risk through uniform exposure, it does not maximize efficiency along the risk–return frontier.

The Monte Carlo Simulation (MCS) approach provided results that fell between EW and MVO. Its stochastic sampling produced portfolios that were more diversified than MVO allocations but generally less efficient in terms of Sharpe ratio. Nonetheless, MCS offers practical robustness: by exploring a large feasible set of portfolios, it accounts for uncertainty in expected returns and covariance structures, making it valuable when parameter estimation is noisy.

Importantly, the quality of LSTM-based stock predictions contributed to the stability of these optimization outcomes. Since portfolio performance relies heavily on input estimates of returns, the relatively low prediction errors of the LSTM model ensured that optimized portfolios reflected realistic market behavior. This synergy between accurate forecasting and optimization highlights the potential of combining machine learning with financial engineering.

E. Summary

The integration of LSTM-driven forecasting models with portfolio optimization techniques demonstrates significant advantages in the domain of quantitative investment management. LSTM networks captured temporal dependencies in stock price data, enabling predictions that were sufficiently robust to inform downstream optimization tasks. This predictive accuracy reduced estimation errors in expected returns, which is a critical factor in ensuring the reliability of portfolio allocations.

Among the optimization approaches, MVO emerged as the most effective, consistently producing the highest Sharpe ratio and thereby delivering the best trade-off between return and risk. Its analytical framework, grounded in modern portfolio theory, allowed it to construct efficient portfolios that outperformed both the naive Equal Weight strategy and the more exploratory Monte Carlo method. While MCS provided diversification benefits and robustness to parameter uncertainty, it did not consistently achieve the efficiency of MVO.

These results collectively reinforce the potential of hybrid approaches that blend deep learning with optimization frameworks. By first leveraging deep learning for accurate return forecasting and subsequently applying optimization techniques for allocation, investors can significantly enhance portfolio performance. The findings suggest that such integrated systems can support data-driven decision-making in dynamic financial markets, offering a valuable tool for both institutional and individual investors. Future work may extend this framework by incorporating additional asset classes, alternative deep learning architectures (e.g., GRUs, Transformers), and practical constraints such as transaction costs and liquidity.

V. Discussions

A. Interpretation of Results

The study shows that combining deep learning forecasts with optimization strategies produces adaptive and efficient portfolio allocations. By leveraging LSTM-generated forecasts, both the Monte Carlo and MVO approaches were able to outperform the Equal-Weight strategy in risk-adjusted terms. This demonstrates that machine learning-driven inputs can significantly enhance traditional financial models, offering investors more robust strategies in dynamic market conditions.

B. Strenghts of the Approach

A major strength of this work lies in its integration of predictive deep learning with portfolio optimization. LSTM, designed to handle sequential and time-dependent data, successfully modeled stock price movements by capturing long-term dependencies in time series. This feature provided more accurate forecasts than simple regression or traditional ARIMA-type models. The framework also offered flexibility by evaluating multiple strategies — EQ, MC, and MVO — allowing for comparative insights and adaptability to different investor risk profiles. Importantly, the system achieved a better risk-return trade-off by directly linking stock-level predictions to portfolio-level outcomes.

C. Why LSTM?

The choice of LSTM is justified by the complex, sequential nature of stock market data. Unlike feed-forward neural networks, LSTM networks are specifically designed to remember patterns over long-time horizons, making them particularly suitable for financial time series, where dependencies may span across weeks or months. Furthermore, their ability to capture non-linear relationships between features such as open, high, low, and volume makes them more powerful than linear forecasting models. Thus, LSTM provided the foundation for generating reliable forecasts that fed into the optimization models.

D. Limitations

While the approach demonstrated promising results, it is not without limitations. The dataset was relatively small, reducing the ability of the LSTM to generalize across diverse market conditions. Portfolio optimization also relied on assumptions such as normally distributed returns and constant covariance, which often fail in real markets, particularly during volatile periods. Additionally, the study excluded macroeconomic, sentiment, and news-based factors that are known to influence stock prices. Finally, Monte Carlo simulations, though powerful, become computationally expensive with larger asset pools, potentially restricting scalability.

E. Improvements

Future work can improve accuracy and robustness by expanding the dataset to include longer time spans and more diverse companies. Feature engineering could incorporate additional variables such as interest rates, market indices, or technical indicators. Ensemble models that combine LSTM with other architectures (e.g., GRU, Transformers) may further enhance predictive strength. Moreover, improving the efficiency of Monte Carlo simulations through parallelization or hybrid optimization methods can make the system more scalable.

F. Future Scope

The framework developed here can be extended in several directions. First, incorporating real-time data feeds and automating rebalancing would make the model suitable for live trading systems. Second, integrating macroeconomic and sentiment analysis (from news or social media) would allow the forecasts to account for external market drivers. Third, the optimization framework could be expanded to include alternative strategies such as risk-parity or Black-Litterman models, providing investors with broader choices. Ultimately, this line of work lays the foundation for AI-driven portfolio management systems that adapt dynamically to changing financial environments, bridging the gap between predictive modeling and practical investment strategies.

References

[1] H. Markowitz, “Portfolio selection,” *Journal of Finance*, vol. 7, no. 1, pp. 77–91, 1952.

[2] R. L. McDonald, *Derivatives Markets*, 4th ed. Pearson, 2019.

[3] P. Glasserman, *Monte Carlo Methods in Financial Engineering*. Springer, 2004.

[4] A. Patel and B. Mehta, “Stock market prediction using support vector regression,” *International Journal of Computer Applications*, vol. 181, no. 1, pp. 45–52, 2018.

[5] S. Shen, H. Jiang, and T. Zhang, “Stock price prediction using random forests and gradient boosting,” *Procedia Computer Science*, vol. 174, pp. 188–195, 2020.

[6] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[7] Z. Zhang, S. Zohren, and S. Roberts, “Deep learning for portfolio optimization,” *arXiv preprint*, arXiv:1810.01273, 2018.

[8] H. Chen and Y. Hao, “A hybrid LSTM-MVO model for portfolio optimization,” *Expert Systems with Applications*, vol. 167, pp. 114384, 2021.

[9] R. D’Ecclesia and C. Magrini, “Monte Carlo-based portfolio selection under deep learning forecasts,” *Journal of Computational Finance*, vol. 25, no. 3, pp. 55–78, 2022.