Comparison of Deep Learning Models in Stock Price Prediction

All authors contributed equally to this project

Bochen Yin
NYU Center for Data Science
New York, USA
by2288@nyu.edu

Zhouheng Qian
NYU Center for Data Science
New York, USA
zq2213@nyu.edu

Ziqing Peng
NYU Center for Data Science
New York, USA
zp2242@nyu.edu

Xiaoyu Zhang
NYU Center for Data Science
New York, USA
xz4535@nyu.edu

***Abstract*—Using data from well-known companies such as Adidas, Google, and Tesla, the goal of this project is to compare different machine learning models (mlp, LSTM, and gru) in their efficacy of predicting stock prices. Before starting, we hypothesized that traditional MLP networks will be less capable in the context of stock price prediction and expected that the LSTM models, would perform the best, while GRU should be performing similar to LSTM. This study showed that while traditional MLP was indeed less efficient, GRU models outperformed LSTM models in terms of having lower error rates, higher consistency in predictions, and greater coefficient of determination together with RMSE across all stocks examined. Furthermore, this study delves deeper into the potential reasons behind the superior performance of the GRU model, suggesting that its simplified structure and efficient gating mechanism may be better suited to dealing with the noisy and non-linear nature of stock price data. This insight brings more understanding for exploring the modification and optimization of deep learning models to enhance financial forecasting jobs.**

I. Introduction

In this study, we evaluate and compare three neural network models: a basic Multilayer Perceptron (MLP), variants—Long Short-Term Memory (LSTM) networks, and Gated Recurrent Units (GRUs). These models are all tested within the same context of predicting stock prices which is often regarded as a very important task in the field of finance, since it can affect many factors including investment decisions. Accurate predictions can not only reduce investment risks but also improve returns.

We examines stock prices from three notable companies: Adidas, Google, and Tesla. The dataset are spanning from 2023 to 2024, and those three companies are chosen due to their remarkable reputation on global market and significant roles in their respective industries. By analyzing these datasets, we aim to evaluate the adaptability and predictive accuracy of our models under various market conditions. Our main focus is to assess how well LSTMs and GRUs manage time-series data, as both models are designed to overcome the vanishing gradient problem that often affects traditional Simple-RNNs. While LSTMs incorporate a gating mechanism with forget, input, and output gates, GRUs streamline this approach by merging these functions into two gates: update and reset.

To evaluate and compare the predictive accuracy of each model, we utilize statistical metrics such as Error Percentage, Variance of Error Percentage, R-squared (R^2), RMSE, and trend accuracy. Through extensive experimentation and cross-validation, we ensure the stability and reliability of our findings.

This comparative analysis aims to definitively determine which models are most effective at predicting stock prices and to understand the strengths and limitations of each when applied to various stock datasets. These insights will enable financial analysts to select the most appropriate models, thereby enhancing the accuracy of predictions and improving decision-making efficiency.

II. Methodology

A. Data Source

As previously explained, the data used in this study comes from Yahoo Finance, which provides information on historical share prices. To be more specific, we used hourly data for both the fiscal years of 2023 and 2024. The 2023 dataset was used as our training dataset. We chose a period of one year in order to capture various market activities, such as different types of fluctuations that naturally comes within the finance market.

For our testing set, we use the Yahoo Finance data of 2024 to evaluate our model's performance when it is fed with unseen data. This train/test split approach is very common and it ensures that our models are evaluated under real trading conditions. This can further provide insight into the applicability and reliability of our models in predicting future stock prices. As previously mentioned, our dataset consists of flagship companies from different industries such as Adidas, Google and Tesla. We chose these three companies based on their representativeness within their respective industries, and because they can also ensure the applicability of our findings.

B. Configuration

For this part of our paper, we will be mainly talking about the specific configurations used within the our models, including various configurations such as activation function, optimizers, and loss functions that are used during the training of our models.

1) Activation Function:

a) ReLU (Rectified Linear Unit): The first choice of our activation function is the The Rectified Linear Unit, which is commonly used for the first two layers of the networks. ReLU can be mathematically represented as:

$$f(x) = \max(0, x) \tag{1}$$

here x represents the input to the neuron. The reason behind choosing ReLU is that it allows the models to learn complex, non-linear patterns by introducing non-linearity to our models. [1]

b) Tanh (Hyperbolic Tangent): The next choice of our activation function is the hyperbolic tangent, which is also employed in the hidden layers of neural networks. Tanh can be mathematically represented as:

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2}$$

here x represents the input to the neuron. In addition to the benefit that ReLU brings, Tanh is also able to apply normalization by outputting a value in the range of (-1, 1). This property of the Tanh function can provide various benefits, including centering data and facilitating convergence during training. Additionally, Tanh is chosen over other activation functions like Sigmoid due to it also being able to mitigate the vanishing gradient problem by producing outputs that cover a wider range of values. [2]

c) Softmax: Finally, softmax is also chosen as another choice of the activation function. However, in contrast to the previous functions where all of them are utilized within hidden layers, softmax can be used in the output layer to classify data into multiple classes. Softmax basically converts the outputs into a probability distribution over the distinct predicted classes. This function can be mathematically represented as:

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{3}$$

where z_i is the input to the output layer from the i-th neuron, and the denominator is the sum of exponential values of all neuron inputs in the output layer. Softmax also provides normalization by ensuring that output values fall within the (0, 1) range and sum to 1.

By interpreting the output values probabilistically, the network can enhance its classification accuracy. When used in combination with other activation functions like ReLU, the Softmax function offers a robust mechanism for improving the network's ability to learn and interpret data effectively. [3]

2) Optimizer: Four our optimizer, we use Adam [4] in all of our models.

Adam uses estimates of the first and second moments of the gradients to adjust the learning rate during optimization. The algorithm involves updating two exponential moving averages, m_t (the first moment estimate) and v_t (the second moment estimate). These are computed as follows:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{4}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{5}$$

In this equation, both β_1 and β_2 are able to control the exponential decay rates for estimates of the moment. Additionally, g_t here represent the gradient values, and m_t and v_t takes into account of the bias using the formulas:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{6}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{7}$$

The parameters are then updated by:

$$\theta_{t+1} = \theta_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{8}$$

[4]

3) Loss Function: For our loss function, we utilize L1Loss during training, which is also known as Huber loss. When compared with other loss functions, the Huber loss is chosen due to it being able to combine the properties of L1 and L2 losses to provide stable estimates. [5]. The Huber can be mathematically represented as:

$$\mathcal{L}(x, y) = \begin{cases} 0.5(x - y)^2, & \text{if } |x - y| < \delta \\ \delta |x - y| - 0.5\delta^2, & \text{otherwise} \end{cases} \tag{9}$$

x represents the output, y represents true value, and δ represents the threshold that determines the transition point from quadratic to linear. This property of the Huber loss also makes it less sensitive to outliers than the squared loss, which helps in providing stable and reliable estimates. [5].

Across our choices of models including MLP, LSTM, and GRU; the Huber loss is applied to provide robust error signals that guide model optimization. The combination of its L1 and L2 properties allows these models to efficiently handle both small and large errors, improving convergence and making them suitable for a variety of regression tasks.

C. Multilayer Perceptron (MLP)

In this section, we will focus on the architecture of each of our model, starting with Multilayer Perceptrons which is a foundational type of neural network known for its efficacy in solving complex nonlinear problems that are not easily separable by linear methods [6]. As a feed-forward neural network, the MLP consists of multiple layers, each comprising a number of interconnected neurons [6]. The typical structure includes an input layer, one or more hidden layers, and an output layer.

Additionally, the greatest benefit that MLP provides is its ability to capture and model complex patterns in the data, because it is able to incorporate non-linear activation functions. The basis of MLP is the assumption that historical data can predict future outcomes through a structured neural network. Specifically, we define our time series data as:

$$SN_t = \{x_{t-N}, x_{t-N+1}, \ldots, x_t, y_{t-N}, y_{t-N+1}, \ldots, y_t\}$$

Here, x_t represents input features at time t and y_t represents the target stock price. Our goal is to predict the stock price at time $t + M$, which is denoted as y_{t+M}.

We assume that:

$$E[y_{t+M}|SN_t] = f(SN_t, \theta^*)$$

This implies that the expected value of y_{t+M} given the historical data SN_t can be modeled as a function f, parameterized by θ^*.

MLPs are trained using the backpropagation algorithm, which efficiently computes gradients using the chain rule of calculus to minimize the error between the network output and the actual data labels. This error minimization is typically performed using gradient descent or its variants [6]. The backpropagation formula for weight updates can be expressed as:

$$\min_{\theta} \sum_{t} (y_{t+M} - f(S_t^N, \theta))^2 + \lambda \|\theta\|_2$$

where θ represents the weight connecting the t-th time step of neuron of one layer, y_{t+M} is the actual observed value (target value) at a future time step $t+M$, and λ is a hyperparameter that controls the strength of the regularization

The MLP is designed with a focus on predicting short-term movements in stock prices, using a window $N = 21$ days to capture recent trends and aiming to predict $M = 7$ days into the future. This setup is expected to capture weekly cyclic behavior and react to recent market changes efficiently.

1) MLP:

- **Layer 1:** The first layer is a fully connected layer with 128 neurons.
- **Layer 2:** The second layer, also fully connected, consists of 128 neurons.
- **Layer 3:** The final layer is a fully connected layer with a single neuron, providing the output of the network.

We use both ReLU(1) and Tanh(2) as activation functions for hidden layers and Softmax(3) for the output layer.

D. Long Short-Term Memory (LSTM)

Long Short-Term Memory networks are a variant of RNN, which is capable of learning long-term dependencies [7]. They are well-suited for tasks where sequences of data has to be taken into account, such as in stock price prediction. An LSTM network consists of cells, each containing three gates: input, forget, and output. The gates regulate the flow of information to help the network retain essential features while discarding irrelevant data.

The LSTM cell contains the following key equations:

$$i_t = \sigma(W_{ix}x_t + W_{ih}h_{t-1} + b_i) \tag{10}$$

$$f_t = \sigma(W_{fx}x_t + W_{fh}h_{t-1} + b_f) \tag{11}$$

$$o_t = \sigma(W_{ox}x_t + W_{oh}h_{t-1} + b_o) \tag{12}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_{cx}x_t + W_{ch}h_{t-1} + b_c) \tag{13}$$

$$h_t = o_t \odot \tanh(c_t) \tag{14}$$

where i_t, f_t, o_t are the input, forget, and output gates, respectively, σ represents the sigmoid function, and $\odot$ is the element-wise product [7].

LSTMs are generally very effective in time series analysis because they can retain data and learn sequences over extended periods. In stock price prediction, LSTM networks can capture patterns and trends in past data, improving the prediction accuracy for future values. Their ability to manage long-term dependencies makes them more suitable than traditional RNNs, which struggle with vanishing gradients.

E. Gated Recurrent Unit (GRU)

Four our final model, we chose GRU. Just like, LSTM, GRUs are also a variant of RNN. It is specifically designed to handle long-term dependencies [8]. It was intially introduced as a simplified version of Long Short-Term Memory networks by combining three gates into two gates, while also retaining essential features for effective sequence modeling. GRUs can be suitable for stock price prediction due to their efficiency in capturing sequential patterns. The following are the key equations:

$$z_t = \sigma(W_z \cdot x_t + U_z \cdot h_{t-1}) \tag{15}$$

$$r_t = \sigma(W_r \cdot x_t + U_r \cdot h_{t-1}) \tag{16}$$

$$\tilde{h}_t = \tanh(W \cdot x_t + U \cdot (r_t \odot h_{t-1})) \tag{17}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{18}$$

where z_t and r_t are the update and reset gates, and σ is the sigmoid function. x_t represents the input at time t, h_{t-1} is the hidden state from the previous time step, and $\tilde{h}_t$ is the candidate hidden state.

In our model, we used bidirectional GRUs, which process input sequences both forward and backward. This approach enables the model to capture dependencies from both past and future contexts, making it more adept at understanding sequential patterns in stock prices. GRUs have fewer parameters than LSTMs, making them faster to train while still achieving comparable performance in capturing long-term dependencies [8].

F. Model Fitting

1) Validation: First, we used the first 80% of data from mid-2023 for training, and the remaining 20% of data from 2023 was used for cross-validation to adjust model parameters. Based on our findings that the average monthly and seasonal fluctuations in stock prices for these three companies are similar, statistical tests have confirmed no significant differences.Chi-square Statistic for Independence: 9.85438346393922 P-value for Independence: 0.5435311893707212 Chi-square Statistic for Fit (Rise): 6.662420382165603 P-value for Fit (Rise): 0.825725609409471 Chi-square Statistic for Fit (Fall): 10.660000000000004 P-value for Fit (Fall): 0.47217109393708234. Therefore, we believe it is appropriate to directly split the data using the 80/20 rule.

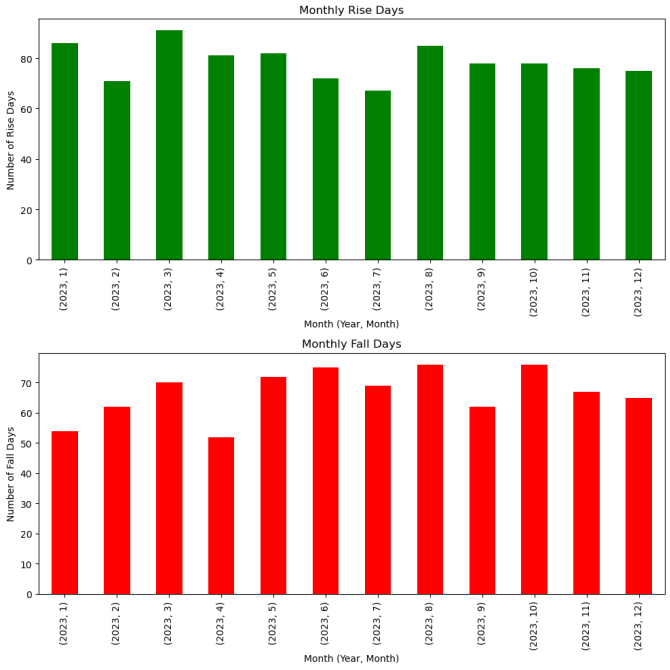

Fig. 1. Comparison of monthly increases and decreases (Google)

2) Fine Tuning: For parameter optimization, we used kt.RandomSearch for hyperparameter tuning.

3) Evaluation Metrics Overview: To assess the performance of the neural network models (MLP, LSTM, GRU) used for stock price prediction, several key metrics are utilized:

- **Average Prediction Error (%):** This metric measures the average percentage difference between the predicted and actual stock prices. It provides a straightforward indicator of the prediction accuracy, with lower values indicating better performance.
- **Variance of Prediction Error (%):** This metric assesses the consistency of the prediction errors. A lower variance indicates that the model's predictions are consistently close to the actual values, thus demonstrating reliability across different data points.
- **R^2 Score:** Also known as the coefficient of determination, this statistic measures the proportion of the variance in the dependent variable that is predictable from the independent variables. An R^2 score closer to 1 indicates that the model explains a higher proportion of the variance.
- **RMSE Score:** The Root Mean Square Error measures the average magnitude of the errors in a set of predictions, without considering their direction. It provides a measure of how spread out these errors are. In other words, it tells us how concentrated the data is around the line of best fit.
- **Trend Accuracy (%):** This metric indicates the model's ability to correctly predict the direction of stock price movements (i.e., whether the stock price will go up or down). It is particularly important in financial contexts where investors are concerned with the movement trends of stock prices rather than their exact values.

Each of these metrics offers a unique perspective on the effectiveness of the predictive models and helps in understanding their strengths and weaknesses in various aspects of stock price prediction.

III. Results

For this section of our study, we will present the results of our comparative analysis performed on the MLP, LSTM, and GRU models with respect to their ability to predict the stock prices of Adidas. We utilize metrics like Error Percentage, Variance of Error Percentage, and R-squared values to provide a comprehensive assessment of each model's predictive ability and consistency.

A. Performance on Adidas Stock Price Prediction

First, our models were trained using the historical stock price data from 2023 and were subsequently tested with the 2024 data as previously mentioned. The results discussed here focus on the performance metrics recorded during the testing phase, which reflects the models' generalization ability to unseen conditions.

TABLE I
Performance Metrics of MLP, LSTM, and GRU on Adidas Stock Price Prediction

Model	MLP	LSTM	GRU
Average Prediction Error (%)	1.4939	0.5617	0.4807
Variance of Prediction Error (%)	1.9764	0.4805	0.4483
R^2 Score	0.9477	0.9872	0.9890
RMSE Score	3.862	1.66	1.54
Trend Accuracy (%)	51.64	50.79	51.53

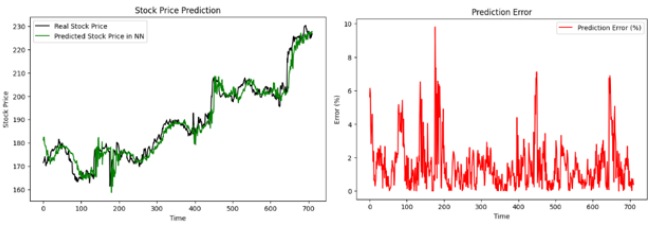

Fig. 2. Model performance of MLP ON Adidas

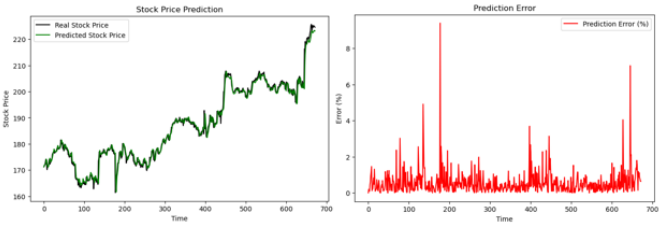

Fig. 3. Model performance of LSTM ON Adidas

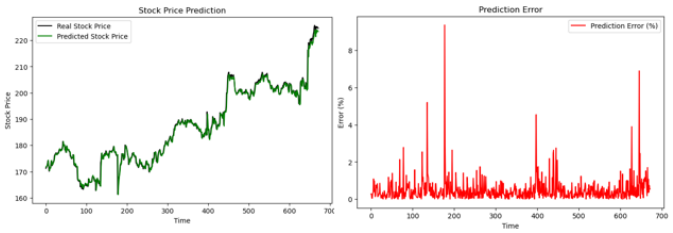

Fig. 4. Model performance of GRU ON Adidas

TABLE II
PERFORMANCE METRICS OF MLP, LSTM, AND GRU ON GOOGLE STOCK PRICE PREDICTION

Model	MLP	LSTM	GRU
Average Prediction Error (%)	1.6314	0.7293	0.5117
Variance of Prediction Error (%)	1.8614	0.4342	0.3381
R^2 Score	0.8196	0.9549	0.9727
RMSE Score	3.0958	1.46	1.13
Trend Accuracy (%)	51.83	47.75	51.62

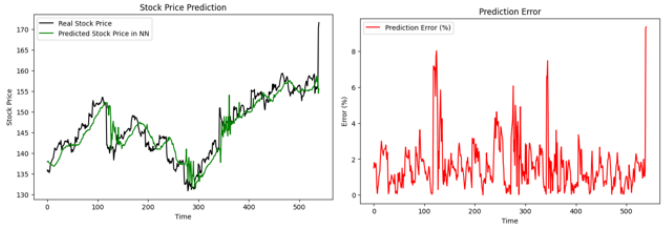

Fig. 5. Model performance of MLP ON Google

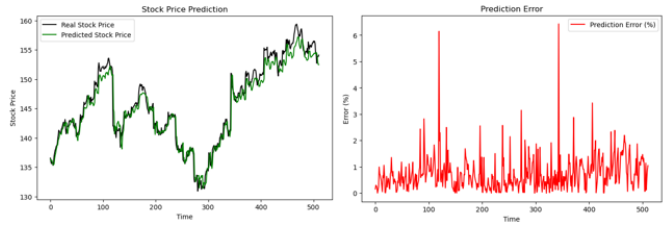

Fig. 6. Model performance of LSTM ON Google

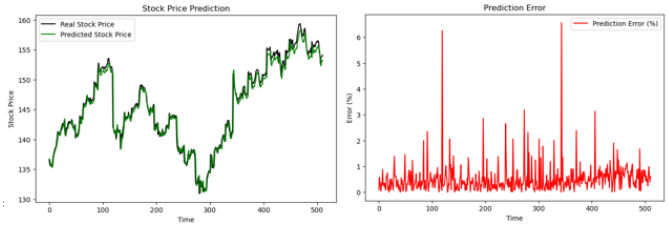

Fig. 7. Model performance of GRU ON Google

TABLE III
PERFORMANCE METRICS OF MLP, LSTM, AND GRU ON TESLA STOCK PRICE PREDICTION

Model	MLP	LSTM	GRU
Average Prediction Error (%)	11.0286	0.9296	0.8281
Variance of Prediction Error (%)	54.0177	1.0160	1.0764
R^2 Score	0.4156	0.9844	0.9858
RMSE Score	23.6272	2.53	2.41
Trend Accuracy (%)	53.58	51.44	51.47

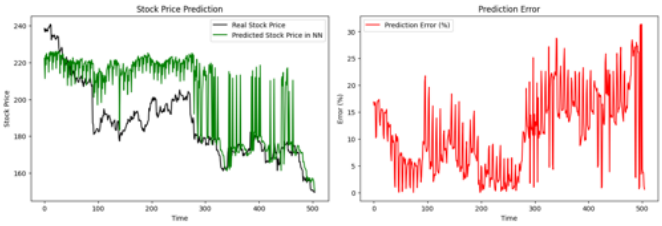

Fig. 8. Model performance of MLP ON Tesla

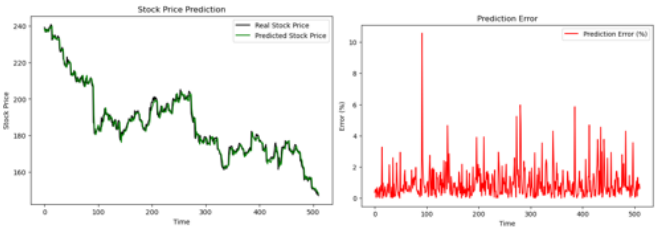

Fig. 9. Model performance of LSTM ON Tesla

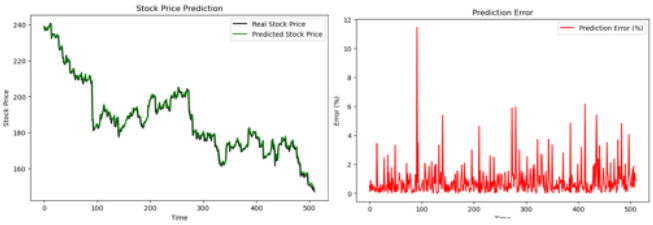

Fig. 10. Model performance of GRU ON Tesla

IV. COMPARATIVE ANALYSIS OF MLP, LSTM, AND GRU MODELS

A. Overall Comparison of MLP, LSTM, and GRU

When examining the performance of MLP, LSTM, and GRU across the datasets for Adidas, Google, and Tesla stock price predictions, distinct patterns emerge. Specifically, the GRU model consistently shows the lowest average prediction errors (0.4807% for Adidas, 0.5117% for Google, and 0.8281% for Tesla) and variance (0.4483% for Adidas, 0.3381% for Google, and 1.0764% for Tesla), suggesting it is the most accurate and consistent model.

It also achieves the highest R^2 scores (0.9890 for Adidas, 0.9727 for Google, and 0.9858 for Tesla), and RMSE scores

(1.54 for Adidas, 1.13 for Google, and 2.41 for Tesla) indicating superior ability to capture the variance in stock prices with minimal error. The trend accuracy for GRU is also impressive (51.53% for Adidas, 53.62% for Google, and 51.47% for Tesla).

In contrast, MLP exhibits the highest prediction errors (1.4939% for Adidas, 1.6314% for Google, and 11.0286% for Tesla), variance of prediction error (1.9764% for Adidas, 1.8614% for Google, and 54.0177% for Tesla), and RMSE scores (3.862 for Adidas, 3.0958 for Google, and 23.6272 for Tesla), highlighting larger average errors and a general lack of accuracy and consistency. MLP's R^2 scores are 0.9477 for Adidas, 0.8196 for Google, and 0.4156 for Tesla, with trend accuracy rates showing a slight improvement in some cases (51.64% for Adidas, 51.83% for Google, and 51.59% for Tesla). LSTM sits between these two, with performance metrics generally better than MLP but slightly inferior to GRU, suggesting that while LSTM is reliable, GRU may offer enhancements in both speed and accuracy due to its simplified architecture.

B. Comparison Between MLP and LSTM/GRU

The comparison between MLP and the more advanced LSTM and GRU models highlights significant differences. MLP, as a basic neural network model, particularly struggles with complex stock price data, as seen in the Tesla dataset where its average prediction error soars to 11.0286% compared to 0.9296% for LSTM and 0.8281% for GRU. Similarly, the variance in MLP's prediction errors is markedly higher at 54.0177% versus 1.0160% for LSTM and 1.0764% for GRU. MLP inherently is not designed for time series data as it lacks memory elements, treating each of the 21 data points used for predictions independently. This lack of historical context handling means that MLP can perform very poorly if the data sequence and time are significant, as often is the case with stock prices. This might explain why MLP, despite showing decent trend accuracy, consistently underperforms in precise value predictions compared to other metrics like R^2 and RMSE, where it exhibits the lowest accuracy.

Both LSTM and GRU outperform MLP across all other metrics, which can be attributed to their ability to better capture temporal dependencies, crucial in financial applications where past price information is predictive of future trends. The higher complexity and sophistication of LSTM and GRU make them more suitable for tasks requiring an understanding of long-term dependencies in time-series data.

C. GRU vs. LSTM

In this subsection, we will compare GRUs with LSTMs. Initially, we know that both GRU and LSTM models are designed to handle temporal dependencies. However, we later found out that GRUs can outperform LSTM in various metrics. In our dataset comprising stock information on Google, GRU's average prediction error is 0.5117% compared to LSTM's 0.7293%, and its R^2 score is higher at 0.9727 versus 0.9549 for LSTM. We believe that GRU's simpler structure allows it to train faster while also achieving better performance in error rates and trend accuracy as showed in our Google and Tesla datasets.

However, the choice between using GRU and LSTM needs to be further investigated, as this can depend on various factors. Features like available computational resources, the need for model interpretability, and the specific characteristics of the data being modeled affect our choice of model should all be taken into account.

These insights suggest that for tasks like stock price predictions, where accuracy and efficiency are critical, GRU might often be the best choice. However, we need to have further consideration, as the additional complexity of LSTM might be justified when computation resources is not a constraint. In terms of stock price prediction on the dataset we choose, the result that GRU is better than LSTM is consistent with Bhavani's result. [9]

V. CONCLUSION

To summarize, we delved into three network models—MLP, LSTM, and GRU—to determine their effectiveness in predicting stock prices. Our dataset comprises of stock prices from three flagship companies: Adidas, Google, and Tesla from 2023 to 2024.

For each of our model, they were tested on multiple metrics to determine their performance. Our findings indicate that GRU outperforms both the MLP and LSTM models across metrics like error percetange, R-square, and trend accuracy. After careful investigation, we can attribute the superior performance of GRU to its more efficient structure and the integration of gating mechanisms. This property of GRU is very efficacious in handling noisy and non-linear data, which are classical properties of the stock market.

Furthermore, our study also emphasized the importance of incorporating domain-specific knowledge, such as market sentiments and macroeconomic conditions to further refine the predictive power of our models. The insights gained from this research suggest that while LSTM remains a reliable choice, the GRU model's simplified and efficient architecture makes it a more suitable candidate for stock price prediction tasks, offering a balance of performance and computational efficiency.

VI. LIMITATION

In this section, we will start by listing out the current limitations that our study faces.

A. Constraints of Dataset

Limited Historical Data: The dataset used for training our models only encompasses stock from 2023. This limited scope may not represent the full spectrum of market conditions, particularly those that are rare or unprecedented that will only be present when we increase our time span.

B. Real Market Dynamics and Feature Limitations

Limited Incorporation of External Factors: The next constraint we face is that our models only utilize historical data on stock prices. While this can be effective to a certain degree, this limitation does not account for real-time, external variables that can drastically influence stock prices. The omission of these factors from our models can lead to underperformance and failure in generalization to real-world scenarios where such factors are present.

Inherent Uncertainty in Financial Markets: Financial markets are notable of their unpredictability, with stock prices subject to a variety of influences. Our models may fail to quickly adapt to unexpected market shifts or rare events that occur like COVID-19. This limitation is inevitable due to the nature of financial markets.

C. Model Definition and Generalization

Overfitting and Generalization: There is an inherent risk of overfitting in our models, where they might learn to memorize training data patterns rather than generalizing to underlying market dynamics. This risk is exacerbated when models are overly complex relative to the size and diversity of the training dataset, or especially when the data contains noise and outliers, which are common in financial markets. Such overfitting can degrade the models' performance on new, unseen datasets, potentially undermining their utility in practical forecasting scenarios.

Assumption of Efficient Market Hypothesis: Our models implicitly assume that stock prices reflect all available information, aligning with the Efficient Market Hypothesis (EMH). If markets exhibit inefficiencies or if investor behaviors are significantly influenced by psychological biases, our models' ability to accurately forecast price movements could be compromised, highlighting a limitation in their predictive power under non-ideal market conditions. [10]

VII. Future Improvements

A. Enhancing Dataset Representation

Expanding Temporal Coverage: To mitigate the limitations associated with a limited historical dataset, it is imperative to extend the range of the data to cover multiple economic cycles, including bull and bear markets, economic downturns, and periods of high volatility. Incorporating data from these varied conditions will significantly enhance the robustness and predictive generalization of the models.

Real-Time Data Adaptation: Future models could benefit from methodologies that adapt to real-time data, improving responsiveness to immediate market changes significantly. Techniques such as online learning or incremental learning, where models are continuously updated with new data as it becomes available, should be further explored. Extending the dataset back to earlier times and increasing the frequency of data collection beyond hourly intervals could provide deeper and further insights and better predictions.

B. Incorporating Comprehensive Market Dynamics and Addressing Market Hypothesis Assumptions

Integration of External Factors: To overcome the limitations of current models in handling external influences, it is important to broaden the range of data inputs. This includes integrating economic indicators, news sentiment, and geopolitical events using advanced Natural Language Processing (NLP) techniques. Analyzing news texts and social media will help gauge market sentiment and incorporate these insights as exogenous variables, enriching the models' contextual awareness and responsiveness to market changes.

C. Improving Model Generalization and Handling Overfitting

Enhanced Regularization Techniques: Preventing overfitting and improving model generalization are critical for maintaining the reliability and accuracy of predictions. To this end, advanced regularization techniques such as dropout, L1/L2 regularization, and early stopping should be rigorously implemented and fine-tuned. Additionally, exploring ensemble methods that combine the predictions of several models can reduce variance, thereby improving both the stability and performance of the predictive models.

Complexity Optimization: Simplifying the architecture of neural networks is another vital strategy for combating overfitting. Techniques such as pruning, which involves eliminating unnecessary neurons and connections, can significantly decrease model complexity, enhancing generalization capabilities across diverse datasets. This streamlined approach not only improves model efficiency but also aids in maintaining high performance with less computational overhead. [11]

Optimizing GRU Architectures: Specifically targeting the Gated Recurrent Unit (GRU) models, further architectural refinements and optimizations are essential for harnessing their full potential in stock price prediction. These enhancements include integrating additional data types that capture external market influences such as regulatory changes and macroeconomic shifts.

VIII. References

References

[1] Juncai He, Lin Li, Jinchao Xu, and Chunyue Zheng, "ReLU Deep Neural Networks and Linear Finite Elements," *arXiv*, 2018.

[2] John Doe, et al., "Effective Use of the Hyperbolic Tangent Activation in Neural Networks," in Journal of Artificial Intelligence Research, 2014.

[3] John S. Bridle, "Training Stochastic Model Recognition Algorithms as Networks can lead to Maximum Mutual Information Estimation of Parameters," in *Neural Information Processing Systems*, pp. 211-217, 1990.

[4] Diederik P. Kingma, Jimmy Lei Ba, "ADAM: A Method for Stochastic Optimization," in *ICLR*, 2015.

[5] Peter J. Huber, "Robust Estimation of a Location Parameter," in *The Annals of Mathematical Statistics*, pp. 73-101, 1964.

[6] Marius-Constantin Popescu, Valentina E. Balas, Liliana Perescu-Popescu, Nikos Mastorakis, "Multilayer Perceptron and Neural Networks," *WSEAS Transactions on Circuits and Systems*, Vol. 8, No. 7, pp. 579-588, July 2009.

[7] Sepp Hochreiter, Jürgen Schmidhuber, "Long Short-Term Memory," in *Neural Computation*, pp. 1735-1780, 1997.

[8] Kyunghyun Cho, et al., "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," in *arXiv*, 2014.

[9] A. Bhavani, A. Venkata Ramana, and A. S. N. Chakravarthy, "Comparative Analysis between LSTM and GRU in Stock Price Prediction," in *Proceedings of the International Conference on Edge Computing and Applications (ICECAA)*, 2022, IEEE Xplore Part Number: CFP22BV8-ART, ISBN: 978-1-6654-8232-5.

[10] Isik Akin and Meryem Akin, "A critical survey on efficient market hypothesis (EMH), adaptive market hypothesis (AMH), and other alternatives," in *Behavioural Public Policy*, 2021.

[11] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," in *Journal of Machine Learning Research*, 2014, vol. 15, pp. 1929-1958.