Proceedings of CONF-MLA 2025 Symposium: Intelligent Systems and Automation: AI Models, IoT, and Robotic Algorithms
DOI: 10.54254/2755-2721/2025.LD29855

Configuration Effects on LSTM-Based Stock Prediction: A Case Study on Coca-Cola

Yiyi Yin

Tufts University, Boston, USA
brennayiyiyin49@gmail.com

Abstract. Long Short-Term Memory (LSTM) has long been a popular prediction model across multiple fields, and extensive research has been conducted on its applications. Among these studies, most employ the traditional train–test split ratio for experiments. However, in the field of stock prediction, this ratio-based split is not practical. From an investment perspective, a key consideration lies in determining the optimal length of historical data required to produce robust and accurate predictions for future market periods. To address the practical utilization of prediction models, this study examines how external parameters—such as time window, training size, and test size (in terms of time length)—affect the performance of the LSTM model in predicting the stock price of The Coca-Cola Company. The dataset includes the historical Open, Low, High, and Close prices from January 1, 2000, to December 31, 2024, with the Close price serving as the target variable. Model performances under different parameter settings are compared using Mean Squared Error (MSE) and prediction plots. The experimental results provide insights into how these external parameters influence the LSTM's performance in stock prediction.

Keywords: Long Short-Term Memory, Stock Price Prediction, Time Series Forecasting, Hyperparameter Optimization

1. Introduction

As a specialized form of Recurrent Neural Network (RNN), the Long Short-Term Memory (LSTM) model has been applied in multiple fields, such as language modeling and rhythm learning. Its popularity stems from its ability to address the vanishing gradient problem, which often limits the effectiveness of traditional RNNs [1]. Because LSTM networks can retain and transfer both short-term and long-term dependencies during training, they are considered highly suitable for time series prediction, particularly in financial markets.

Although LSTM has become one of the most widely used time series models in machine learning, its application in real-world financial prediction still requires further investigation. Existing studies have largely focused on comparing LSTM with other models or demonstrating its general utility across domains. In contrast, this paper examines how variations in the training process affect the accuracy of stock prediction in a single-case setting. Specifically, the study investigates practical factors such as the training window size and the length of the training period, which are closely related to real-world applications.

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2025.LD29855

All experiments are conducted on Coca-Cola (KO) stock, selected for its relatively stable trading patterns, making it a suitable case for study. Model performance is evaluated using mean squared error (MSE), and prediction accuracy is further illustrated through comparative plots. This research is grounded in the following literature.

2. Literature review

2.1. LSTM as a prediction tool in multiple financial market

In recent years, Long Short-Term Memory (LSTM) networks have become a popular tool for time series prediction in financial markets. However, LSTM is not the only option for such tasks, and it has frequently been compared with both traditional statistical models and other modern machine learning methods.

One commonly used benchmark is the Autoregressive Integrated Moving Average (ARIMA) model, which remains advantageous in terms of interpretability and short-term forecasting accuracy. For example, Zhang (2023) [2] conducted experiments on five representative global stock indices from 2015 to 2022, comparing ARIMA and LSTM in terms of prediction accuracy, mean squared error (MSE), and root mean squared error (RMSE). The results showed that LSTM outperformed ARIMA, largely due to its ability to better capture nonlinear fluctuations in financial time series. Similarly, Holm and Åkesson (2024) [3] compared ARIMA and LSTM using the closing prices of the Stockholm market index (OMXS30), with training window sizes of 7, 30, and 60 days. Their findings indicated that model performance depended on the chosen time horizon: ARIMA and LSTM performed similarly with a 7-day window, LSTM outperformed ARIMA with a 30-day window, while ARIMA yielded better results in the 60-day window.

Another algorithm frequently compared with LSTM is the Gated Recurrent Unit (GRU), a simplified variant of LSTM. Saputra et al. (2021) [4] investigated Ethereum (ETH) price prediction using both LSTM and GRU over a five-year dataset, tuning hyperparameters for optimal performance. Their evaluation, based on RMSE, MAE, MAPE, and R^2 scores, found no significant differences, although GRU showed slightly better performance. A more comprehensive comparison was conducted by Barua et al. (2024) [5], who evaluated RNN, LSTM, CNN, GRU, and Attention-LSTM for predicting stock prices of five major Indian companies. Using MAE, MSE, RMSE, and R^2 as evaluation metrics, they found that CNN and GRU performed better for relatively stable time series, while Attention-LSTM was more effective for volatile series. LSTM, however, still exhibited advantages in specific cases, highlighting the context-dependent nature of these models.

Collectively, these comparative studies suggest that while LSTM remains a strong candidate for financial time series forecasting, its performance—like that of other deep learning models—depends heavily on dataset characteristics and configuration choices. Therefore, further case studies are necessary to better understand how to optimize these models in practical applications.

2.2. Parameter configuration in LSTM model

To better understand how LSTM should be utilized in different contexts, many studies have examined the influence of training parameters on model performance under real-world conditions.

For example, Zhang and Pinsky (2025) [6] compared the predictive performance of LSTM on the S&P 500 and Nasdaq Composite Index using different training periods of three months, six months, and one year. With all other conditions held constant, their experiments with a 10-day prediction window indicated that the one-year, close-to-close model achieved higher accuracy for the S&P 500,

DOI: 10.54254/2755-2721/2025.LD29855

while the three-month, close-to-open model performed better for the Nasdaq. The authors attributed this difference to the higher volatility of the Nasdaq, which favored a shorter training period.

Similarly, Kobiela et al. (2022) [7] conducted a detailed comparison between LSTM and ARIMA on Nasdaq stock data, varying the number of input features and prediction horizons from 30 days to nine months. Their results, based on mean absolute percentage error (MAPE), showed that ARIMA performed better as the forecasting horizon lengthened, whereas LSTM had advantages in shorter-term predictions.

These studies highlight the importance of parameter configuration in specific applications. Factors influencing model performance include internal hyperparameters, such as batch size and the number of hidden layers, as well as external parameters, such as training dataset size, training window length, and feature selection. Each factor can affect prediction accuracy differently depending on the dataset and market conditions. Despite theoretical guidance, practical experiments remain essential to fully understand their implications. In this paper, we focus specifically on external parameters—particularly training length and time window size—and evaluate how these choices affect LSTM performance in stock price prediction.

2.3. LSTM – extensions and improvements

Although this study focuses on the standard LSTM model, numerous works have sought to enhance LSTM performance through architectural modifications or optimization techniques.

For instance, Sebastian and Tantia (2025) [8] incorporated an attention mechanism into LSTM layers, enabling the model to better capture relevant temporal dependencies. Gülmez (2024) [9] integrated the Sand Cat Swarm Optimization (SCSO) algorithm with LSTM to optimize model parameters, achieving improved forecasting accuracy. Similarly, Al Theeb, Smadi, and Al-Qaydeh (2025) [10] proposed a novel hyperparameter tuning strategy by combining Bayesian Optimization (BO) with LSTM and GRU architectures.

These studies demonstrate that enhanced or hybrid versions of LSTM often outperform the original model in specific applications. However, they also suggest that there remains considerable scope for further exploration and refinement of LSTM-based models in financial prediction.

The experiment conducted in this paper is built as described below.

3. Methodology

3.1. Model

In this paper, we focus exclusively on the standard LSTM model. As a specialized form of RNN, LSTM shares the same basic network structure as a traditional RNN (see Figure 1; Olah, 2015) [11].

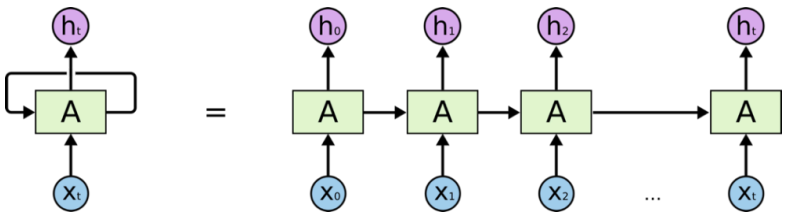

Figure 1. RNN layer structure. Reprinted from understanding LSTM networks, by C. Olah, 2015, Colah's blog (https://colah.github.io/posts/2015-08-Understanding-LSTMs/)

DOI: 10.54254/2755-2721/2025.LD29855

Based on the illustration in Figure 1, an RNN consists of input layers x_t, hidden layers A, and output layer h_t. RNNs differ from traditional neural networks in that their hidden layers are "connected," allowing information to pass from one time step to the next. This structure makes RNNs particularly suitable for tasks that require sequential awareness, such as time series prediction, context prediction, and rhythm generation. However, as noted in the introduction, RNNs suffer from the vanishing gradient problem, which limits their ability to maintain long-term memory. LSTM addresses this issue by incorporating a gating mechanism into its hidden layers. While retaining the basic RNN structure, LSTM introduces an additional state that stores long-term memory and a set of gates that control which information should be retained as long-term memory (see Figures 2 and 3; Olah, 2015) [11].

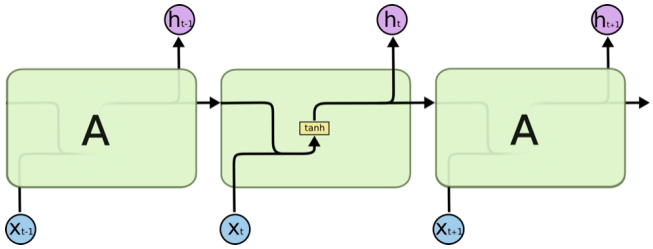

Figure 2. Traditional RNN contains only a single layer. Reprinted from understanding LSTM networks, by C. Olah, 2015, Colah's blog (https://colah.github.io/posts/2015-08-Understanding-LSTMs/)

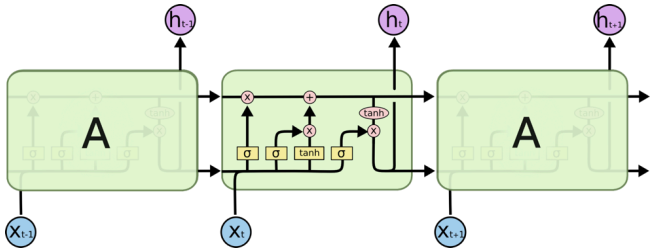

Figure 3. LSTM contains one more extra layer to store the long-term information and passing it through the states. Reprinted from understanding LSTM networks, by C. Olah, 2015, Colah's blog (https://colah.github.io/posts/2015-08-Understanding-LSTMs/)

Within the LSTM hidden layer, each interaction can be interpreted as a question: "How much information should pass through?" In the diagrams, "how much" corresponds to the gates, represented by sigmoid transformations, while "information" is processed through tanh functions to facilitate effective transformations. The LSTM hidden layer contains three gates in total (see Figure 4; Olah, 2015) [11], along with an additional memory cell C_t.

DOI: 10.54254/2755-2721/2025.LD29855

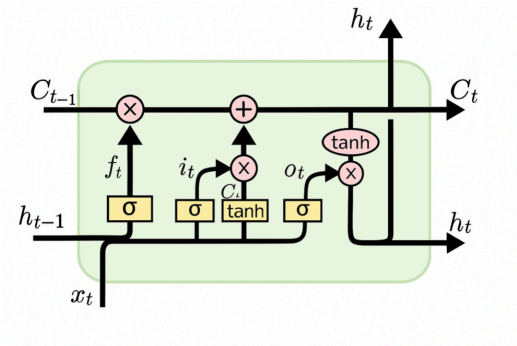

Figure 4. LSTM hidden layer structure with three gates presented as sigmoid functions in the graph. Adapted from understanding LSTM networks, by C. Olah, 2015, Colah's blog (https://colah.github.io/posts/2015-08-Understanding-LSTMs/)

"Forget gate" decides how much long-term information should be passing through previous state to the current state. It is related to the current input $\boldsymbol{x_t}$ together with the output $\boldsymbol{h_{t-1}}$ from former inputs, thus its function presented as:

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

To update the $\boldsymbol{C_{t-1}}$ to $\boldsymbol{C_t}$, information from the current input $\boldsymbol{x_t}$ should also contribute part of the information to the long-term memory. Here comes the "input gate", which controls how much information from current state should be passed to the current long-term state. It is related to the current input $\boldsymbol{x_t}$ together and the output $\boldsymbol{h_{t-1}}$ from the former inputs as well, thus it could be presented as:

$$\mathbf{i_t} = \sigma\left(\mathbf{W_i} \cdot [\mathbf{h_{t-1}}, \ \mathbf{x_t}] + \mathbf{b_i}\right) \tag{2}$$

The information that may pass through the input gate are output $\boldsymbol{h_{t-1}}$ from previous inputs and current input $\boldsymbol{x_t}$. The information passes through input gate could therefore be presented as:

$$\mathbf{\widetilde{C}_t} = \mathbf{tanh}\left(\mathbf{W_c} \cdot [\mathbf{h_{t-1}}, \ \mathbf{x_t}] + \mathbf{b_c}\right) \tag{3}$$

Recall the forget gate decides how much information that would pass from the previous long-term state $\boldsymbol{C_{t-1}}$ to the current long-term state $\boldsymbol{C_t}$, the $\boldsymbol{C_t}$ update equation could be written as:

$$C_t = f_t \odot C_{t-1} + i_t \odot \widetilde{C}_t \tag{4}$$

The output of current state $\boldsymbol{h_t}$ needs to be updates as well. An "output gate" has been set for deciding how much information should be passed from the current long-term state $\boldsymbol{C_t}$. The output gate again is related to the last output $\boldsymbol{h_{t-1}}$ and current input $\boldsymbol{x_t}$:

DOI: 10.54254/2755-2721/2025.LD29855

$$\mathbf{o_t} = \sigma\left(\mathbf{W_o} \cdot [\mathbf{h_{t-1}},\ \mathbf{x_t}] + \mathbf{b_o}\right) \tag{5}$$

Thus, the current output $\boldsymbol{h_t}$ is:

$$\mathbf{h_t} = \mathbf{o_t} \odot \mathbf{tanh C_t} \tag{6}$$

By using this architecture, LSTM forms an additional memory layer that efficiently mitigates the vanishing gradient problem. This paper specifically investigates how external parameters affect prediction accuracy. The experimental setup is described in the next section.

3.2. Experiment

This experiment is based on the standard form of LSTM. The code was adapted from a PyTorch repository on GitHub (Hichenway, n.d.) [12]. All experiments used the same hyperparameters: batch size of 64, hidden layer size of 128, number of hidden layers as 2, and a dropout rate of 0.2. This consistency ensures that the comparisons are not affected by the model's internal hyperparameters. The loss function used in this study is mean squared error (MSE). Each training process was run to completion with a learning rate of 0.001 and an early stopping mechanism with a patience of 5, meaning the training would terminate early if no significant improvement occurred over five consecutive epochs.

The dataset consists of daily historical stock prices of The Coca-Cola Company (KO) on the New York Stock Exchange (NYSE), downloaded from Yahoo Finance (2025) [13], covering the period from January 1, 2000, to December 31, 2024. The original dataset includes multiple features: Open, Close, High, Low, Adjusted Close, and Volume. Adjusted Close was excluded due to its similarity to Close, and Volume was excluded due to its limited informational value for this specific case. Four features—Open, Close, High, and Low—were selected as input features, with the Close price as the target variable. The Close price was chosen due to its practical relevance in financial analysis. A summary of the dataset is provided in Table 1, and a complete line graph of the monthly Close prices is shown in Figure 5. As illustrated in Figure 5, KO stock prices exhibit a generally stable pattern, making them suitable for prediction tasks.

Table 1. Data summary for KO stock price

Price	Close	High	Low	Open
count	6288	6288	6288	6288
mean	27.5341	27.7285	27.3271	27.5309
std	15.5674	15.6612	15.4723	15.5699
min	9.4907	9.6264	9.4753	9.4984
25%	13.3754	13.5166	13.2253	13.3745
50%	24.8136	24.9466	24.6234	24.8169
75%	36.9379	37.2087	36.7222	36.9720
max	70.3956	70.8970	69.6628	70.6752

DOI: 10.54254/2755-2721/2025.LD29855



Figure 5. KO close price daily (2000 to 2024)

This experiment is designed to investigate how external parameters affect LSTM model predictions. This topic is considered valuable due to its practical significance. In a more theoretical setting, training and test sets are typically split by proportion, such as 80/20 or 90/10. However, in the context of the stock market, this approach is less meaningful. From a financial perspective, investors are more concerned with how much historical information is needed to predict future trends or, given the current data, how far into the future the model can reliably forecast. This perspective motivates a nontraditional approach to train-test splitting, based on time periods rather than proportions. In this experiment, different training periods (6 months, 12 months, 18 months, 24 months, 36 months) and test periods (3 months, 6 months, 12 months) were evaluated to gain practical insights into model performance. The experimental results are presented in the following section.

4. Result

Four different training periods were applied to the test set—6 months, 12 months, 18 months, and 24 months—to compare model performance. Two timesteps, representing the number of previous days used to predict the next one-day result, were tested: 10 days and 20 days. Prediction plots and MSE results for the 6-month case are shown in Figure 6 and Table 2.

The MSE table provides an overall view of model performance. For a 10-day timestep, the best result was achieved with a 12-month training period. For a 20-day timestep, the best result corresponded to a 48-month training period. Combining these results with the prediction plots, the 12-month training period with a 10-day timestep outperformed all other cases, while the 36-month training period with a 20-day timestep performed the worst. For a 6-month prediction horizon, the 10-day timestep generally produced better results than the 20-day timestep, and shorter training periods tended to yield superior performance.

The plots indicate that, although prediction accuracy is limited in some cases, most models successfully capture the overall trends of the data. However, all models struggled to predict rapid changes, such as peaks in the time series.

DOI: 10.54254/2755-2721/2025.LD29855

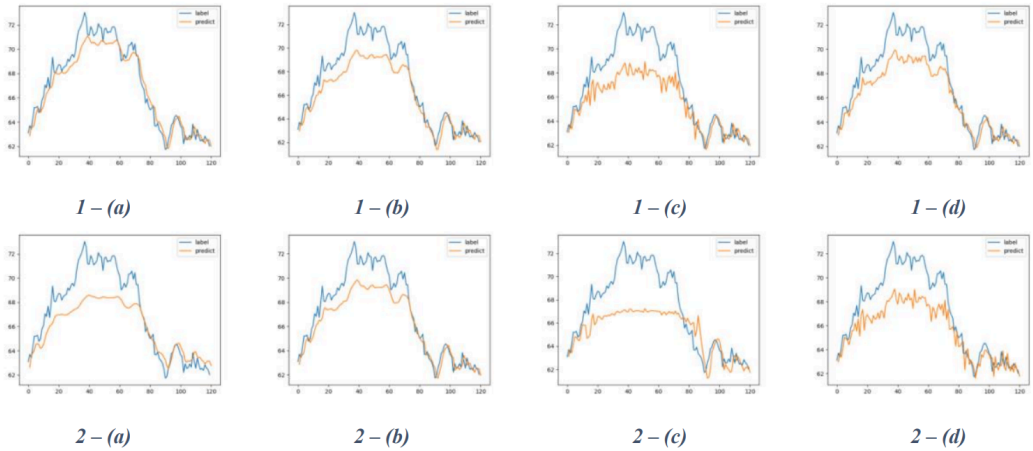

Figure 6. Prediction results for 6-month period with different training length and timestep. (X-axis as Days Y-axis as Close Stock Price) panels 1 (a–d) use a 10-day time step; panels 2 (a–d) use a 20-day time step; sub-panels (a–d) represent training-data ranges of 12, 24, 36, and 48 months, respectively

Table 2. MSE comparison for 6-month period with different training length and timestep. (In each column the best result is marked as red while the worst is marked as black)

Train Data Range for 6m Test	MSE with Timestep = 10d	MSE with Timestep = 20d
12	0.0393	0.1983
24	0.1558	0.1477
36	0.2874	0.4708
48	0.0742	0.1449

A longer test period of 12 months was also evaluated. The results are presented in Figure 7 and summarized in Table 3.

When the test period was extended from 6 months to 12 months, the best performance was achieved with the longest training period of 48 months, while the worst performance occurred with a 24-month training period, for both the 10-day and 20-day timesteps. Based on the prediction plots, it is noteworthy that, in the 12-month test period, there is no significant difference in prediction accuracy between the 10-day and 20-day timesteps.

Overall, even with a 12-month test period, most models were able to capture the general trends of the data. While the models continued to struggle with rapid changes in the time series, their performance was generally better than that observed in the 6-month prediction tests.

DOI: 10.54254/2755-2721/2025.LD29855

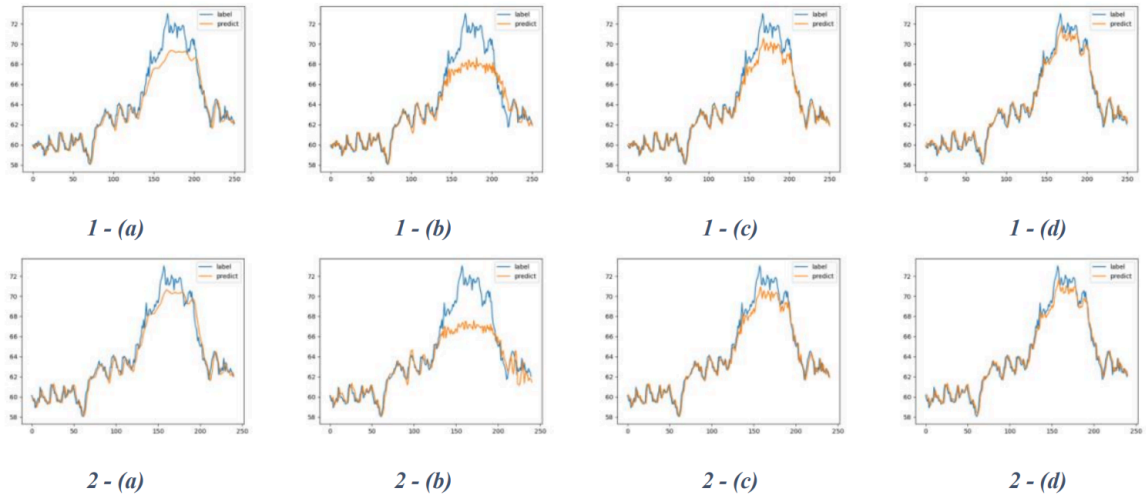

Figure 7. Prediction results for 12-month period with different training length and timestep. (X-axis as Days Y-axis as Close Stock Price) Panels 1 (a–d) use a 10-day time step; panels 2 (a–d) use a 20-day time step; sub-panels (a–d) represent training-data ranges of 12, 24, 36, and 48 months, respectively

Table 3. MSE comparison for 12-month period with different training length and timestep. (In each column the best result is marked as red while the worst is marked as black)

Train Data Range for 12m Test	MSE with Timestep = 10d	MSE with Timestep = 20d
12	0.0814	0.0422
24	0.1876	0.2937
36	0.0370	0.0272
48	0.0099	0.0108

An extreme case was also considered in this experiment, simulating a scenario in which the available dataset is extremely limited. For this case, a 3-month test period was used, and different training periods—6 months, 12 months, 18 months, and 24 months—were compared. This setup reflects the practical situation of a newly listed company entering the stock market, where only a short historical record of stock prices is available. Given the short duration of the 3-month test period, only a 10-day timestep was applied in this experiment. The results are presented as prediction plots in Figure 8, with the corresponding MSE values summarized in Table 4.

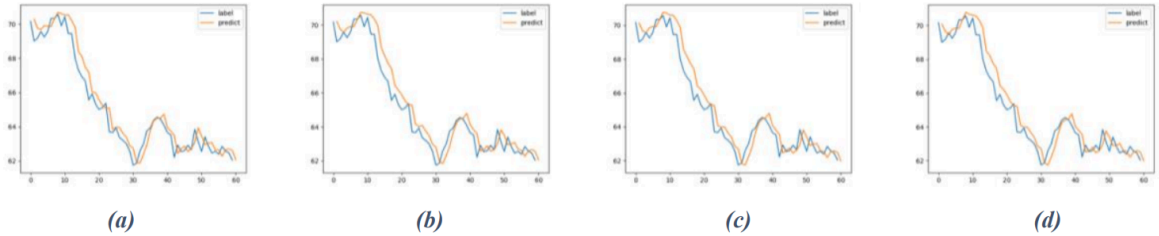

Figure 8. Prediction results for 3-month period with different training length and timestep. (X-axis as Days Y-axis as Close Stock Price) Panels (a) to (d) correspond to training data ranges of 6 months, 12 months, 18 months, and 24 months, respectively

DOI: 10.54254/2755-2721/2025.LD29855

Table 4. MSE comparison for 3-month period with different training length. (The best result is marked as red while the worst is marked as black)

Train Data Range for 12m Test	MSE with Timestep = 10d
12	0.0309
24	0.0253
36	0.0306
48	0.0346

In this extreme case, the differences among both the MSE results and the prediction plots were minimal. The 24-month training model slightly outperformed the others, while the 48-month model performed slightly worse. According to the plots, all models successfully captured the general trends of the stock prices.

5. Discussion

5.1. Analysis

Comparing the results for the 6-month and 12-month prediction periods shows that the length of the training period can have different effects on LSTM prediction performance. For the 6-month test range (Figure 6 and Table 2), predictions tended to perform better with shorter timesteps and shorter training periods. This aligns with general understanding of short-term stock behavior, where stock prices rely more on short-term memory. Longer training periods may introduce more noise and increase the risk of overfitting in such cases.

For the 12-month test range (Figure 7 and Table 3), the results were different. Here, timestep had less impact on prediction performance, and longer training periods generally performed better. This is likely because a longer test period requires more historical context, which necessitates a longer training dataset. Since the model relies more on long-term memory in this scenario, the influence of timestep becomes less significant.

The observed patterns generally follow this trend, although not exactly. Unusual peaks during the test period likely contributed to deviations, indicating that standard LSTM models struggle to predict rapid changes in time series data.

In the extreme case with very limited data, different training periods were applied to a 3-month prediction horizon. The training period length had minimal effect, and predictions showed no significant differences. This suggests that when training data are extremely limited, the length of the training period becomes less important. However, compared to the 6-month and 12-month cases (Figures 6 and 7), the 3-month test period excluded rapid-change periods, making it easier for the model to capture future trends.

5.2. Limitations & thoughts

This study explored how the length of the training period affects LSTM predictions under different circumstances, providing valuable insights but also highlighting several limitations.

The experiment was designed based on the practical assumption that investors train models based on a time range rather than a traditional train-test split. However, the experiment could be more informative if conducted using data from a company's initial stock market period. Due to data

DOI: 10.54254/2755-2721/2025.LD29855

accessibility and for comparability, Coca-Cola's stock history was used. More realistic results may be obtained using start-up stock data.

Additionally, the results were affected by unusual peaks in the data, reflecting a limitation of the standard LSTM in handling rapid changes. This may explain why some results deviated from general expectations and highlights the need to consider potential rapid fluctuations when tuning external parameters for time series prediction using LSTM.

6. Conclusion

This paper examined the effects of external parameters—training data range, test data range, and timestep—on LSTM predictions of Coca-Cola's closing stock prices, with a focus on real-world applicability. Model performance was evaluated using prediction plots and MSE values. The results indicate that shorter test periods perform better with shorter timesteps and shorter training periods, while longer test periods benefit from longer training periods, with timestep having less impact. In extreme cases with very limited data, the training period length had minimal effect on prediction accuracy. The results were also influenced by LSTM's limitations in handling rapid changes in stock prices. Future research should investigate LSTM's ability to predict unusual patterns and rapid fluctuations to improve its applicability in financial time series forecasting.

References

[1] Sepp Hochreiter, Jürgen Schmidhuber; Long Short-Term Memory. Neural Comput 1997; 9 (8): 1735–1780. doi: https: //doi.org/10.1162/neco.1997.9.8.1735
[2] Zhang, Z. (2023). Comparison of LSTM and ARIMA in price forecasting: Evidence from five indexes. In Proceedings of the 2023 2nd International Conference on Economics, Smart Finance and Contemporary Trade (ESFCT 2023) (pp. 40–46). Atlantis Press. https: //doi.org/10.2991/978-94-6463-268-2_6
[3] Holm, A., & Åkesson, A. (2024). A comparison of LSTM and ARIMA forecasting of the stock market.
[4] Saputra, M.P., Ibrahim, R.A., & Saputra, R.S. (2025). Comparative Analysis of LSTM and GRU Models for Ethereum (ETH) Price Prediction. International Journal of Business, Economics, and Social Development.
[5] Barua, M., Kumar, T., Raj, K., & Roy, A. M. (2024). Comparative Analysis of Deep Learning Models for Stock Price Prediction in the Indian Market. FinTech, 3(4), 551-568. https: //doi.org/10.3390/fintech3040029
[6] Zhang, X., & Pinsky, E. (2025). S& P-500 vs. Nasdaq-100 price movement prediction with LSTM for different daily periods. Machine Learning with Applications, 19, 100617. https: //doi.org/10.1016/j.mlwa.2024.100617
[7] Kobiela, D., Krefta, D., Król, W., & Weichbroth, P. (2022). ARIMA vs LSTM on NASDAQ stock exchange data. Procedia Computer Science, 207, 3836–3845. https: //doi.org/10.1016/j.procs.2022.09.445
[8] Sebastian, A., & Tantia, V. (2025). Multi-variate LSTM with attention mechanism for the Indian stock market. International Journal of Information Management Data Insights, 5(2), 100350. https: //doi.org/10.1016/j.jjimei.2025.100350
[9] Gülmez, B. (2024). Stock price prediction using the Sand Cat Swarm Optimization and an improved deep long short-term memory network. Borsa Istanbul Review, 24(Suppl. 2), 32–46. https: //doi.org/10.1016/j.bir.2024.12.002
[10] AL Theeb, N., Smadi, H., & Al-qaydeh, N. (2025). Bayesian-optimized ensemble deep learning models for demand forecasting in the volatile situations: A case study of grocery demand during Covid-19 outbreaks. Journal of Industrial Engineering and Management, 18(1), 193-213. doi: https: //doi.org/10.3926/jiem.6571
[11] Diagram illustrating the gate structure of an LSTM unit. Adapted from Understanding LSTM Networks, by C. Olah, 2015, Colah's Blog. https: //colah.github.io/posts/2015-08-Understanding-LSTMs/
[12] Hichenway. (n.d.). stock_predict_with_LSTM [Source code]. GitHub. https: //github.com/hichenway/stock_predict_with_LSTM
[13] Yahoo Finance. (2025). The Coca-Cola Company (KO) stock historical data [Data set]. Yahoo Finance. Retrieved October 7, 2025, from https: //finance.yahoo.com/quote/KO/history