A Long Short-Term Memory (LSTM) Neural Architecture for Presaging Stock Prices

Tej Nileshkumar Doshi, Shubham Ghadge, Yamini Gonuguntla, Namirah Imtieaz Shaik, Ashutosh Mathore and Bonaventure Chidube Molokwu[a]

Department of Computer Science, College of Engineering and Computer Science, California State University, Sacramento, U.S.A.

Keywords: Stock Prediction, LSTM, Deep Learning, Financial Time Series, Machine Learning, Forecasting, Time Series Analysis.

Abstract: Stock price prediction is crucial for informed investment decisions(Bathla, 2020; Hochreiter and Schmidhuber, 1997). This study explores the application of Long Short-Term Memory (LSTM) architecture for analyzing and predicting stock prices of major technology companies: Alphabet Inc. (GOOG), Apple Inc. (AAPL), NVIDIA Corporation (NVDA), Meta Platforms, Inc. (META), and Tesla Inc. (TSLA). The fundamental challenge addressed is capturing temporal dependencies and complex patterns in financial time series data, which traditional statistical methods often fail to model accurately(Box et al., 1978; Hyndman and Athanasopoulos, 2013). Our methodology involved collecting historical stock data from Yahoo Finance API(Edwards et al., 2018), preprocessing through normalization and sequence creation(Hochreiter and Schmidhuber, 1997), and training separate LSTM models for each stock. Results indicate that LSTM models provide satisfactory accuracy with R^2 scores exceeding 0.93 for most stocks(Li et al., 2023; Selvin et al., 2017), capturing both short-term and long-term patterns(Panchal et al., 2024; Ouf et al., 2024). The implications are significant for investors and financial analysts seeking enhanced predictive tools for market forecasting(Pramod and Pm, 2020).

1 INTRODUCTION

The stock market represents one of the most dynamic and complex financial systems, characterized by rapid and unpredictable price movements influenced by macroeconomic indicators, market sentiment, geopolitical events, and social media trends (Edwards et al., 2018; Li et al., 2023; Ouf et al., 2024). Traditional statistical models like ARIMA and GARCH, while theoretically sound, often assume stationarity and linearity, limiting their effectiveness in capturing the non-linear, volatile nature of real-world stock data (Box et al., 1978; Hyndman and Athanasopoulos, 2013).

The emergence of Machine Learning and Deep Learning has enabled more sophisticated forecasting algorithms capable of capturing hidden patterns in sequential data. Long Short-Term Memory (LSTM) networks, a specialized form of Recurrent Neural Networks, have demonstrated outstanding performance in modeling temporal sequences by overcoming the vanishing gradient problem through their gating mechanisms (Hochreiter and Schmidhuber, 1997; Selvin et al., 2017; Panchal et al., 2024).

This research develops and evaluates an LSTM-based model for stock price prediction using historical data from five major technology companies, comparing performance against traditional and alternative deep learning approaches using metrics like RMSE, MAE, and R^2 Score (Bathla, 2020; Li et al., 2023; Xiao et al., 2024).

2 LITERATURE REVIEW

Recent studies consistently demonstrate LSTM's superiority over traditional approaches for stock prediction (Panchal et al., 2024; Bathla, 2020; Selvin et al., 2017). (Panchal et al., 2024) showed LSTM models achieving lower RMSE values compared to ARIMA across Apple, Google, and Tesla datasets,

[a] https://orcid.org/0000-0003-4370-705X

Doshi, T. N., Ghadge, S., Gonuguntla, Y., Shaik, N. I., Mathore, A. and Molokwu, B. C.
A Long Short-Term Memory (LSTM) Neural Architecture for Presaging Stock Prices.
DOI: 10.5220/0013691900003985

In *Proceedings of the 21st International Conference on Web Information Systems and Technologies (WEBIST 2025)*, pages 473-480
ISBN: 978-989-758-772-6; ISSN: 2184-3252

while ARIMA failed to capture non-linear dependencies. (Xiao et al., 2024) compared LSTM, GRU, and Transformer networks on Tesla stock prediction, concluding LSTM offered the best trade-off between complexity and accuracy.

(Li et al., 2023) applied LSTM models to technology stocks including Apple and Nvidia, demonstrating high accuracy over decade-long price histories and emphasizing LSTM's robustness in capturing long-term temporal dependencies. (Ouf et al., 2024) incorporated Twitter sentiment features into LSTM models, reporting improved accuracy compared to price-only models. These findings, supported by (Selvin et al., 2017; Bathla, 2020; Pramod and Pm, 2020), provide strong motivation for applying LSTM networks to a broader dataset of five major technology stocks.

3 MATERIALS AND METHODOLOGY

3.1 Data Collection and Preprocessing

Historical stock price data for AAPL, GOOG, META, NVDA, and TSLA was collected using the Yahoo Finance API(Edwards et al., 2018; de Prado, 2018), covering the period from January 1, 2012, to December 21, 2022. The size of the dataset is represented as a tuple $(13805, 7)$ where $13,805$ indicates the number of rows and 7 indicates the number of columns. Each company has 2761 rows of data. The data includes features like Open, High, Low, Close, and Volume fields(Edwards et al., 2018).

3.2 Feature Engineering and Data Transformation

100-Day Moving Average (MA100) and 200-Day Moving Average (MA200) were computed to provide additional trend information beyond daily volatility. These technical indicators help smooth short-term fluctuations and highlight longer-term trends, with MA100 capturing medium-term patterns and MA200 focusing on long-term market sentiment(Box et al., 1978; Hyndman and Athanasopoulos, 2013).

Rather than feeding raw prices directly into the model, a sliding window approach was applied to create sequences. For each stock, sequences of the past 60 days of 'Close' prices were prepared to predict the 61st day's 'Close' price. A sliding window approach was adopted to generate the input-output pairs required by the LSTM model.

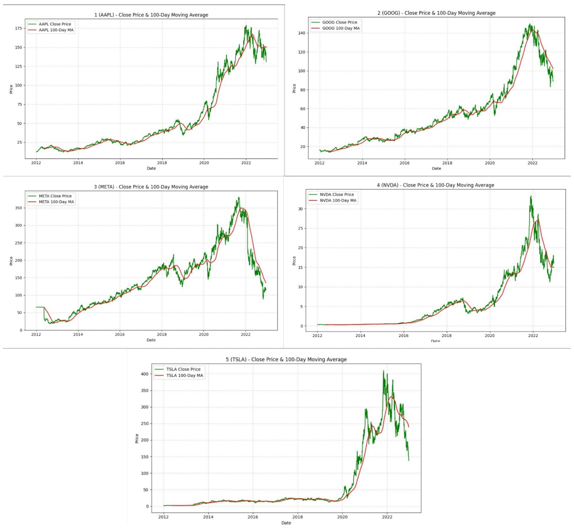

Figure 1: 100-Day MA vs Close Price.

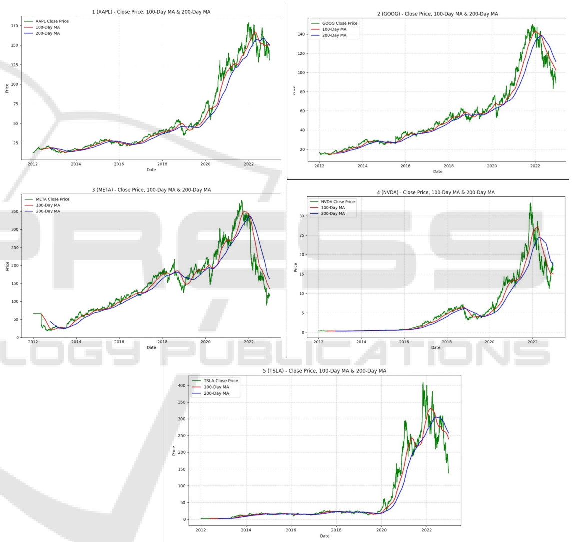

Figure 2: 200-Day MA vs 100-Day MA vs Close Price.

We decided to split the data for our model as follows:

- 80% for Training
- 10% for Validation (from within training set)
- 20% for Testing

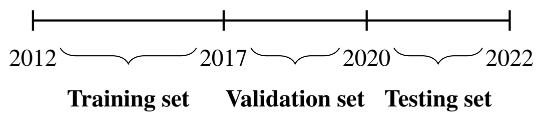

Figure 3: Chronological data split into training, validation, and testing sets.

Fig. 3. above indicates the data split into three sets. Chronological order was preserved to main-

tain the time-dependent structure of the stock price data. After sequence generation, the training, validation, and testing sets were reshaped into a 3D structure required by LSTM layers: (shape) = (samples,timesteps,features)

Where:

- samples = number of sequences,
- timesteps = 60 (days of history),
- features = 1 ('Close' price only).

Standard dense layers expect 2D input, but LSTMs require 3D to maintain temporal relationships across timesteps.

3.3 Working of LSTM Networks

Long Short-Term Memory (LSTM) networks are an extension of Recurrent Neural Networks (RNNs), specifically designed to overcome the vanishing gradient problem and capture long-range dependencies in sequential data.

Each LSTM unit is composed of three primary gates:

- **Forget Gate:** Decides what information from the previous cell state should be discarded.
- **Input Gate:** Determines which new information should be added to the cell state.
- **Output Gate:** Controls the information that will be outputted from the current cell.

At each time step, the LSTM cell takes the current input and the hidden state from the previous time step, processes them through the three gates, updates the internal cell state, and produces a new hidden state.

Mathematically, the gates are computed as follows:

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

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

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{3}$$

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

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

$$h_t = o_t * \tanh(C_t) \tag{6}$$

where:

- σ denotes the sigmoid activation function,
- tanh denotes the hyperbolic tangent activation function,
- W and b represent the weight matrices and bias vectors, respectively,
- h_{t-1} is the hidden state from the previous time step,

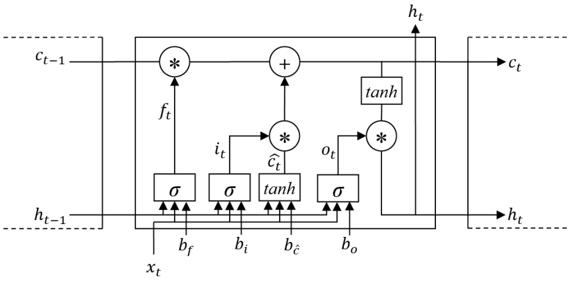

Figure 4: LSTM Architecture(Molokwu and Kobti, 2019).

- x_t is the input at the current time step t.

These mechanisms enable LSTMs to selectively remember or forget information over long sequences, making them ideal for modeling time series data such as stock prices.

Fig. 4. illustrates the internal structure of an LSTM (Long Short-Term Memory) cell. The LSTM unit is composed of four primary components: the Forget Gate, the Input Gate, the Cell State Update, and the Output Gate.

- The Forget Gate determines which parts of the previous cell state (C_{t-1}) should be discarded based on the previous hidden state (h_{t-1}) and the current input (x_t).
- The Input Gate decides which new information should be written into the cell state, again using the current input and the previous hidden state.
- The Cell State Update combines the output of the forget and input gates to update the internal cell memory (C_t).
- The Output Gate generates the new hidden state (h_t) based on the updated cell state and regulates what information will be passed to the next time step.

The inputs to the LSTM cell are the previous hidden state h_{t-1}, the previous cell state C_{t-1}, and the current input x_t. The outputs are the updated cell state C_t and the new hidden state h_t. This structure enables the LSTM to effectively capture both long-term and short-term dependencies in sequential data.

3.4 Model Architecture

The LSTM model was designed with a Sequential architecture:

- LSTM Layer 1: 128 units, return sequences, Input Shape: (60 timesteps, 1 feature). The first LSTM layer processes each 60-day sequence and learns intricate temporal dependencies across multiple trading days. Setting return sequences=True ensures that the output of this layer is a full sequence (not a single value), which is passed to the next LSTM layer for deeper pattern extraction. This layer acts as a feature extractor for sequential data, capturing both short-term and medium-term trends.

- Dropout Layer 1: 0.3. A Dropout layer is inserted after the first LSTM to randomly deactivate 30% of the neurons during each training batch. This technique prevents the model from overfitting by ensuring it does not become too dependent on specific neurons. Dropout regularization encourages generalization over training data.
- LSTM Layer 2: 64 units, no return sequences. The second LSTM layer receives the sequence output from the first LSTM but collapses it into a single vector output by setting return sequences=False. This vector summarizes the temporal features extracted from the entire 60-day input window. This layer acts as a temporal summarizer, enabling the model to output a single future price prediction.
- Dropout Layer 2: 0.3. Another Dropout layer is added to reduce overfitting risks even after feature summarization, ensuring robustness in the learned representations.
- Dense Output Layer: 1 neuron/unit. The final layer is a Dense (Fully Connected) layer with one output neuron. It maps the processed sequential features into a single scalar value — the predicted closing stock price for the next day.No activation function is applied because stock price prediction is a regression task.

3.5 Model Compilation, Training, and Prediction

After defining the LSTM model architecture, the model was compiled using:

- Optimizer: Adam The Adam optimizer was chosen due to its adaptive learning rate capabilities, fast convergence properties, and widespread success in deep learning tasks. Adam combines the advantages of both RMSProp and SGD optimizers, making it well-suited for noisy and sparse datasets like financial time series.
- Loss Function: Mean Squared Error (MSE) MSE is the standard loss function for regression tasks where the goal is to predict continuous values, such as stock closing prices. It penalizes larger errors more heavily, encouraging the model to produce highly accurate predictions.

The model compilation stage ensures that the appropriate optimization and evaluation settings are prepared before training begins.

Next, the training was performed using the training set, with validation on a separate validation set to monitor the model's generalization ability and detect any signs of overfitting.

- Number of Epochs: 100
- Batch Size: 32 samples per training step

During training, the model learned to minimize the MSE loss on the training data while maintaining stable validation loss to ensure good generalization to unseen data.

The training process produced a history object containing loss and validation loss values for each epoch, which were later used for plotting learning curves.

Finally, after the model completed training, predictions were generated on:

- Training set (for internal performance assessment),
- Validation set (to monitor overfitting),
- Testing set (to evaluate real-world generalization).

Since the model was trained on normalized (scaled) data between [0,1], the predicted outputs needed to be inverse transformed back to their original price scale for meaningful interpretation. Inverse transformation was performed using the previously fitted MinMaxScaler on both:

- Predicted values (train, validation, test sets),
- True target values (ground truth for train, validation, test sets).

This rescaling ensures that evaluation metrics like RMSE, MAE, and R^2 Score, and graphical visualizations (actual vs predicted prices) are reported in real stock price units (e.g., 150,300).

3.6 Frameworks, Programming Languages, and Libraries

The proposed LSTM-based stock price prediction system was implemented using the following software tools and libraries:

- **Programming Language:** Python 3.8+
- **Deep Learning Framework:** TensorFlow 2.x (with Keras API)
- **Machine Learning Utilities:** scikit-learn
- **Financial Data Access:** yfinance
- **Data Manipulation:** pandas and NumPy
- **Visualization:** matplotlib

4 EXPERIMENT AND RESULTS

This section presents the outcomes of training the LSTM models, analyzing the learning behavior, evaluating the predictive performance using standard metrics.

4.1 Evaluation Metrics

Model performance was assessed using four standard regression metrics:

- Mean Squared Error (MSE): Measures the average squared difference between actual and predicted values.
- Root Mean Squared Error (RMSE)(Kumar et al., 2021): Square root of MSE, interpretable in the same units as stock price.
- Mean Absolute Error (MAE): Average of absolute differences between actual and predicted values.
- Coefficient of Determination (R² Score): Measures how well predictions approximate real data (1 = perfect prediction).

The Table 1 below shows the evaluation metrics for each of the companies.

Table 1: Model Performance Metrics for Different Companies.

Company	MSE	RMSE	MAE	R^2 Score
Apple	11.73	3.42	2.68	0.9576
Google	25.32	5.03	4.33	0.9314
Meta	78.86	8.88	6.61	0.9877
Nvidia	6.43	2.54	2.19	0.7741
Tesla	204.77	14.31	10.61	0.9367

4.2 Learning Curves

Fig. 5. presents the validation loss across all five companies over 100 training epochs. The validation loss serves as an indicator of the model's ability to generalize to unseen data. Stocks like Google and Apple showed consistent and low validation losses, suggesting stable and effective training. On the other hand, Meta and Tesla experienced more fluctuation in validation loss, particularly during early epochs. This instability reflects the inherent volatility of these stocks in the given timeframe, making prediction more challenging. Nevertheless, all models showed convergence trends, and none displayed sustained overfitting behavior.

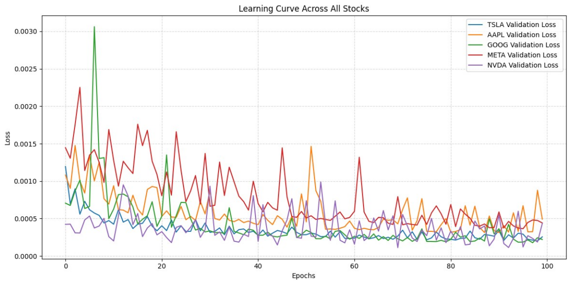

Figure 5: Learning Curves for 5 Companies.

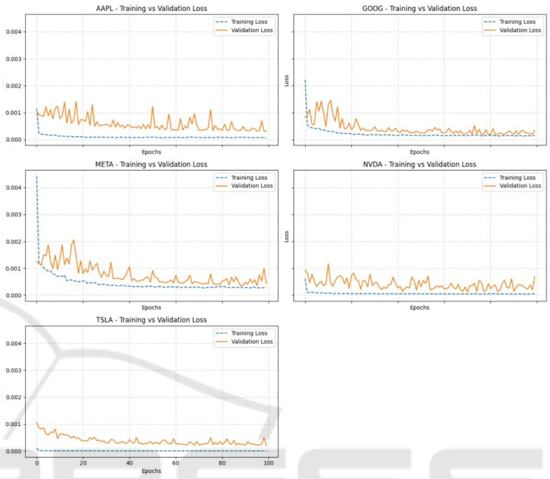

Figure 6: Training Loss vs Validation Loss.

4.3 Training and Validation Loss Analysis

To better understand the learning behavior of the LSTM model, both Training Loss and Validation Loss were recorded during training for each stock. The curves are presented in the Fig. 6. for the five companies individually.

The analysis of the curves revealed the following observations:

- For most stocks (Apple, Google, Meta), the training and validation losses decreased steadily and remained close to each other, indicating that the model generalized well without significant overfitting.
- Nvidia exhibited slightly slower convergence initially, but validation loss ultimately stabilized after around 50 epochs.
- Tesla's stock showed higher validation loss fluctuations during early epochs, which can be attributed to Tesla's high volatility in the real stock market during the studied period (especially 2020-2022).

Table 2: RMSE Comparison of Our LSTM Model with Other Approaches from Literature.

Company	Our LSTM	[1] LSTM	[1] ARIMA	[2] LSTM	[2] GRU	[2] Transformer	[3] LSTM
Apple	3.42	5.52	13.60	N/A	N/A	N/A	18.89
Google	5.03	4.66	7.15	N/A	N/A	N/A	18.89
Meta	8.88	N/A	N/A	N/A	N/A	N/A	N/A
Nvidia	2.54	N/A	N/A	N/A	N/A	N/A	N/A
Tesla	14.31	12.38	23.98	26.09	18.44	16.40	N/A

Table 3: Comparison of R^2 Scores Between Our LSTM Model and Results from (Ouf et al., 2024).

Company	Our LSTM	[4] LSTM (Sentiment Analysis)	[4] LSTM (No Sentiment Analysis)
Apple	95.76%	91%	98%
Google	93.14%	88%	95%
Meta	98.77%	N/A	N/A
Nvidia	77.41%	N/A	N/A
Tesla	93.67%	73%	55%

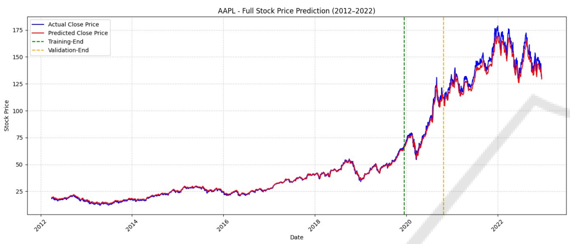

Figure 7: LSTM model prediction graph of Apple.

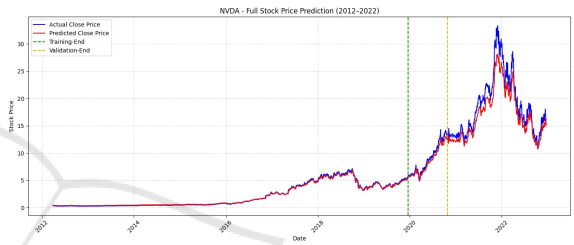

Figure 10: LSTM model prediction graph of Nvidia.

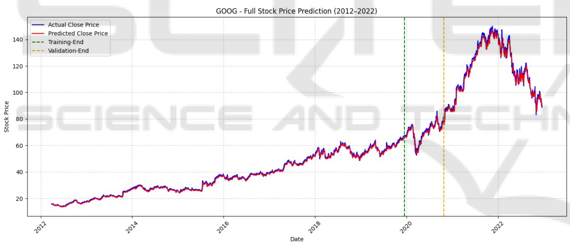

Figure 8: LSTM model prediction graph of Google.

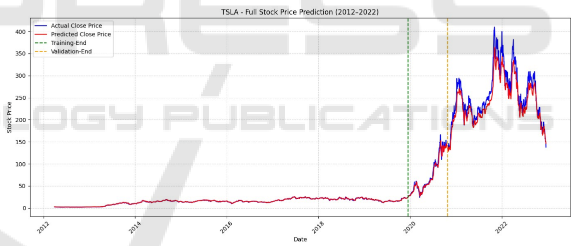

Figure 11: LSTM model prediction graph of Tesla.

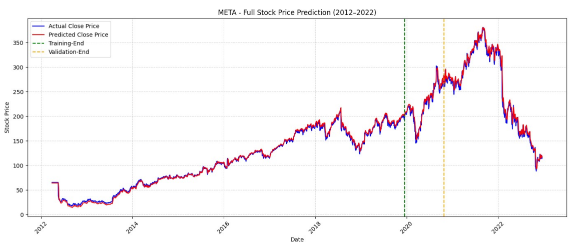

Figure 9: LSTM model prediction graph of Meta.

4.4 Full Time Predictions

The full timeline plots comparing actual and predicted closing prices for each stock across the training, validation, and testing periods revealed several key insights:

- We see in Fig. 7. Fig. 8. Fig. 9. for Apple, Google, and Meta stocks respectively, demonstrated excellent alignment between actual and predicted prices across the entire 2012–2022 timeline, with minimal deviation during both stable and volatile periods.
- In Fig. 10. Nvidia's stock showed good general predictive performance but had slight underfitting during rapid market changes, suggesting that additional features (such as market news sentiment) could further enhance the model for highly volatile stocks.
- Tesla's stock in Fig. 11, due to its extremely high volatility in 2020–2022 (e.g., stock splits, market sentiment shifts), exhibited larger prediction errors during sudden spikes. However, the LSTM model was still able to capture the overall upward trend and major turning points reasonably well.

4.5 Comparative Result Analysis

To benchmark our proposed LSTM model, we compared our experimental results in Table 2 and Table 3 against four other published IEEE papers:

In all cases, the LSTM model successfully learned the temporal dependencies in the stock price sequences and was able to generalize well to unseen testing data.

Importantly, no significant divergence was observed between actual and predicted prices toward the end of the testing period for any of the stocks, confirming the model's stability and robustness.

Furthermore, the prediction plots clearly illustrated that the LSTM model was not merely memorizing training data but was genuinely learning underlying patterns and trends that could be generalized across time periods.

5 DISCUSSION AND LIMITATIONS

The experimental results demonstrate LSTM's high effectiveness for stock price forecasting, with R^2 scores exceeding 0.9 for most stocks confirming successful capture of complex temporal dependencies. Comparative analysis showed LSTM consistently outperforming ARIMA and matching or exceeding GRU and Transformer performance. Key limitations include: (1) univariate analysis using only closing prices, missing benefits from technical indicators like RSI, MACD, or sentiment analysis; (2) single-step prediction limiting real-world trading applicability; (3) computational expense compared to simpler models; and (4) performance degradation on highly volatile stocks like Tesla during extreme market conditions. Despite limitations, the study confirms LSTM networks as reliable tools for financial time series modeling, with architecture suitable for algorithmic trading systems, risk modeling, and portfolio optimization strategies.

6 CONCLUSION AND FUTURE WORK

This study demonstrates LSTM neural networks' effectiveness in predicting stock prices using historical closing price data. Key findings include strong generalization (R^2 ¿ 0.93 for most stocks), stable training without overfitting, robust performance across volatility levels, and superior benchmark performance compared to traditional ARIMA and competitive deep learning models. Results confirm LSTM's power in capturing non-linear, time-dependent structures in stock market data. The model's modularity allows easy future extensions and real-world applications in portfolio management and algorithmic trading. Future enhancements could incorporate additional features (volume, volatility indicators, sentiment analysis), extend to multi-step predictions, combine with attention mechanisms, include macroeconomic indicators, deploy real-time systems with continuous retraining, and apply explainability tools for transparent decision-making.

ACKNOWLEDGMENT

The authors would like to sincerely thank Dr. Bonaventure Chidube Molokwu at California State University, Sacramento, for their invaluable guidance, support, and constructive feedback throughout the duration of this project. Their encouragement and insightful comments were instrumental in shaping the methodology, experiments, and analysis presented in this study.

We also acknowledge the availability of open-source financial data APIs such as Yahoo Finance and the use of powerful deep learning libraries like TensorFlow and Scikit-learn, which greatly facilitated the successful completion of this project.

REFERENCES

Bathla, G. (2020). Stock price prediction using lstm and svr. In *2020 Sixth International Conference on Parallel, Distributed and Grid Computing (PDGC)*, pages 211–214. IEEE.

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. (1978). Time series analysis: Forecasting and control. *The Statistician*, 27:265–265.

de Prado, M. M. L. (2018). Advances in financial machine learning: Numerai's tournament (seminar slides). *Electrical Engineering eJournal*.

Edwards, R. D., MaGee, J., and Bassetti, W. H. C. (2018). *Technical analysis of stock trends*. CRC Press, Boca Raton, 11th edition.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Hyndman, R. J. and Athanasopoulos, G. (2013). *Forecasting: principles and practice*. OTexts.

Kumar, A., Alsadoon, A., Prasad, P. W. C., Abdullah, S. H., Rashid, T. A., Pham, D. T. H., and Nguyen, T. Q. V. (2021). Generative adversarial network (gan) and enhanced root mean square error (ermse): deep learning for stock price movement prediction. *Multimedia Tools and Applications*, 81:3995–4013.

Li, Z., Yu, H., Xu, J., Liu, J., and Mo, Y. (2023). Stock market analysis and prediction using lstm: A case study on technology stocks. *Innovations in Applied Engineering and Technology*.

Molokwu, B. C. and Kobti, Z. (2019). Spatial event prediction via multivariate time series analysis of neighboring social units using deep neural networks. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Ouf, S., Hawary, M. E., Aboutabl, A., and Adel, S. (2024). A deep learning-based lstm for stock price prediction using twitter sentiment analysis. *International Journal of Advanced Computer Science and Applications*.

Panchal, S. A., Ferdouse, L., and Sultana, A. (2024). Comparative analysis of arima and lstm models for stock price prediction. In *2024 IEEE/ACIS 27th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pages 240–244. IEEE.

Pramod, B. and Pm, M. S. (2020). Stock price prediction using lstm. *Test Engineering and Management*, 83(3):5246–5251.

Selvin, S., Vinayakumar, R., Gopalakrishnan, E. A., Menon, V. K., and Soman, K. (2017). Stock price prediction using lstm, rnn and cnn-sliding window model. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1643–1647. IEEE.

Xiao, J., Deng, T., and Bi, S. (2024). Comparative analysis of lstm, gru, and transformer models for stock price prediction. In *Proceedings of the International Conference on Digital Economy, Blockchain and Artificial Intelligence*.