Stock Opening and Closing Price Prediction using LSTM and Technical Indicators

M. Surya
Master of Computer Application
St. Francis College
Bengaluru, India

Nazura Javed
Department of Computer Applications
St. Francis College
Bengaluru, India

Farheen Fathima
Master of Computer Application
St. Francis College
Bengaluru, India

ABSTRACT
Stock price prediction remains a challenging problem in financial forecasting due to the nonlinear, noisy, and highly dynamic nature of market behavior, influenced by a complex interplay of economic events, investor sentiment, macroeconomic policies, and hidden temporal patterns. Accurate prediction of stock prices can significantly benefit individual investors, portfolio managers, and financial institutions by enabling more informed decision-making and risk management. However, the stochastic and non-stationary characteristics of financial time series make reliable forecasting extremely difficult using conventional statistical methods. This paper presents a Long Short-Term Memory (LSTM) based multivariate forecasting framework for predicting the next day's opening and closing prices of four major U.S. equities The proposed framework does not solely rely on raw OHLCV (Open, High, Low, Close, Volume) data. It enriches the input feature space through systematic technical analysis, generating 10 additional indicators including Simple Moving Averages at 5, 10, and 20-day windows (MA5, MA10, MA20), Exponential Moving Average (EMA20), Relative Strength Index (RSI), Moving Average Convergence Divergence (MACD) with its signal line, Bollinger Bands (upper and lower), Daily Return, and Volatility. The framework is evaluated using three standard regression metrics: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and the coefficient of determination (R^2). Experimental results across all four stocks demonstrate that the combination of technical indicator feature engineering with LSTM sequence learning produces meaningful forecasting performance.

Keywords
LSTM, stock prediction, technical indicators, time series forecasting, Yahoo Finance, deep learning, financial prediction, multi-output regression.

1. INTRODUCTION
The stock market serves as one of the most important institutions in the global economy, providing a dynamic platform for capital formation, wealth creation, and investment opportunity. Millions of transactions occur daily across exchanges worldwide, driven by a complex web of economic indicators, corporate earnings, geopolitical events, monetary policy decisions, and collective investor behavior. Stock price forecast provides valuable insights for individual investors and financial professional as even small improvements in prediction accuracy can translate into substantial financial gains or risk mitigation.

However, predicting stock prices is inherently difficult because financial markets are nonlinear, non-stationary, and influenced by a multitude of observable and hidden factors. Traditional statistical approaches such as Autoregressive Integrated Moving Average (ARIMA) and vector autoregression models have been widely used for time-series forecasting, but they often rely on assumptions of linearity and stationarity that do not hold in real financial markets. These methods struggle to capture the complex, long-range temporal dependencies and abrupt regime changes that characterize stock price movements. As a result, there has been a growing shift toward machine learning and deep learning techniques that can model these nonlinear relationships more effectively.

Among deep learning architectures, Long Short-Term Memory (LSTM) networks have emerged as a particularly powerful tool for sequential data analysis. Originally introduced to address the vanishing gradient problem in standard Recurrent Neural Networks (RNNs), LSTM networks use gating mechanisms that allow them to selectively remember or forget information across long sequences [1]. This makes them naturally suited for financial time-series data, where current prices depend on patterns spanning days, weeks, or even months. Unlike feedforward neural networks that treat each input independently, LSTM networks maintain a hidden state that captures the temporal context of the entire input sequence, enabling them to learn evolving market dynamics.

Beyond raw price data, technical indicators derived from historical market information have long been used by traders and analysts to identify trends, momentum shifts, and volatility patterns. Indicators such as Moving Averages, Relative Strength Index (RSI), Moving Average Convergence Divergence (MACD), and Bollinger Bands encode domain knowledge about market behavior that raw price data alone cannot convey. Recent research has demonstrated that incorporating these indicators as additional input features can significantly improve the forecasting performance of deep learning models, as they provide the network with richer contextual information about market conditions [2], [3].

Most existing studies focus on predicting a single target variable, typically the closing price. However, financial markets produce multiple related price points each trading day, and predicting them jointly can capture interdependencies that separate models would miss. The opening price reflects overnight sentiment and pre-market activity, while the closing price captures the cumulative effect of intraday trading. Predicting both simultaneously allows the model to learn a shared representation of daily market dynamics, potentially improving consistency and accuracy for both targets. This multi-output approach has gained traction in recent literature, with studies showing that joint prediction frameworks can outperform single-target models [4], [5].

The main objective of this work is to design a practical, end-to-end stock forecasting pipeline that combines systematic

technical indicator engineering with LSTM-based sequence learning to predict next-day Opening and Closing prices. The framework is designed to be transparent, reproducible, and accessible for academic research, while still capturing meaningful market behavior through a carefully designed feature set. The specific contributions of this paper are:

- A complete feature-engineering pipeline that generates 10 technical indicators across three categories (trend, momentum, and volatility) from raw OHLCV data, producing a 16-feature input representation.
- A two-layer stacked LSTM architecture with dropout regularization, designed to capture both low-level temporal patterns and higher-order sequential dependencies in 60-day price windows.
- A multi-output prediction strategy that simultaneously forecasts next-day Open and Close prices within a single unified model, enabling shared representation learning.
- A structured experimental evaluation using MAE, RMSE, and R^2 across four major U.S. equities (AAPL, MSFT, NVDA, AMZN), providing a reproducible benchmark for future comparison.

The remainder of this paper is organized as follows. Section II reviews related work on LSTM-based stock prediction and the use of technical indicators in deep learning models. Section III describes the dataset and the feature engineering process and also presents the methodology, including data preprocessing, normalization, sequence generation, and model architecture. Section IV details the experimental setup.. Section V presents the results and discussion, and Section VI concludes the paper with directions for future work.

2. RELATED WORK

Long Short-Term Memory (LSTM) networks have become the dominant deep learning architecture for stock price forecasting. LSTMs enable networks to learn dependencies across long time horizons [1]. Hence, they are appropriately applied for analyzing sequential financial data, wherein current prices are influenced by patterns spanning days, weeks, or months. The work [2] models LSTM for financial markets and demonstrates that LSTM networks outperform memory-free classification methods including random forests, deep neural networks, and logistic regression for predicting directional movements of S&P 500 stocks. This study established LSTM as a viable tool for quantitative trading strategies.

Many state-of-the art works have explored the combination of technical analysis indicators with deep learning models to improve forecasting performance. [3] proposed an optimal LSTM (O-LSTM) model using adaptive stock technical indicators and a Correlation-Tensor approach for feature selection. this method achieved a mean prediction accuracy of 59.25% across multiple stocks, significantly outperforming benchmark machine learning classifiers. [5] introduced a PCA-LSTM model that applies Principal Component Analysis to reduce the dimensionality of technical indicator features before feeding them into an LSTM network. By extracting principal components from the raw indicator set, this approach reduced data correlation and improved prediction accuracy for stock price fluctuation trends, highlighting the importance of feature preprocessing in technical-indicator-based forecasting.

Researchers have proposed numerous hybrid architectures that combine LSTM with other deep learning components to capture different aspects of financial data, [6] introduced a feature fusion LSTM-CNN model that combines temporal features extracted by LSTM with image features extracted by CNN from stock chart representations. The experiments on S&P 500 ETF data demonstrated that fusing features from different data representations helped to reduce prediction error.

[7] proposed a CNN-LSTM framework that uses three-dimensional CNN input tensors incorporating time series data, technical indicators, and inter-stock correlations. The CNN extracts spatial features while the LSTM captures temporal dependencies. The their experimental results showed superior performance over standalone models for predicting stock price movement direction.

[8] developed a graph-based CNN-LSTM algorithm (SACLSTM) that constructs sequence arrays of historical data and leading indicators (options and futures) as input images for the CNN framework. Feature vectors extracted through convolutional and pooling layers serve as input to the LSTM, achieving improved prediction performance across ten stocks from U.S. and Taiwan markets.

Based on the literature review conducted by us, we find the following research gaps.

- Most studies focus on predicting a single target variable, typically the closing price, while the relationship between opening and closing prices remains underexplored in multi-output frameworks.
- While many studies use technical indicators as features, few provide a systematic comparison of feature engineering pipelines that include trend, momentum, and volatility indicators together.
- Many work emphasize complex hybrid architectures (CNN-LSTM, GRU-CNN, attention mechanisms) at the expense of simpler, more interpretable models that may be sufficient for practical applications.

This paper addresses the research gaps by: (1) proposing a multi-output LSTM framework that jointly predicts next-day Open and Close prices, (2) implementing a comprehensive technical indicator pipeline with 16 features across three categories, and (3) demonstrating that a straightforward two-layer LSTM architecture can achieve competitive performance without the complexity of hybrid models, making it accessible for academic research and practical deployment.

3. METHODOLOGY

This section discusses the complete pipeline including data collection and feature engineering, preprocessing and the architecture of the proposed LSTM model.

3.1 Data Collection

Daily stock records were collected from Yahoo Finance for four major U.S. companies including Apple Inc. (AAPL), Microsoft Corporation (MSFT), NVIDIA Corporation (NVDA), and Amazon.com Inc. (AMZN). Since financial data are time-dependent, the rows are not shuffled during preprocessing or train-test splitting. This preserves the temporal structure necessary for sequence-based models.

3.2 Feature Engineering (Technical Indicators)

The dataset is enriched by generating technical indicators. These indicators help the model capture trend, momentum, and volatility information that raw price data alone cannot provide. Moving averages MA5, MA10, and MA20 help capture short and medium-term trends by smoothing price fluctuations over

5, 10, and 20-day windows respectively. The Exponential Moving Average (EMA20) emphasizes recent prices more strongly than a simple moving average, making it more responsive to recent market changes. The indicators are computed as follows:

- MA_n = (P_1 + P_2 + ... + P_n) / n
- RS = Average Gain / Average Loss
- RSI = 100 - 100 / (1 + RS)
- MACD = EMA12 - EMA26
- Signal = EMA9(MACD)
- Return_t = (Close_t - Close_(t-1)) / Close_(t-1)
- Volatility_t = standard deviation of daily returns over a rolling window

The Relative Strength Index (RSI) identifies momentum and possible overbought or oversold conditions, with values ranging from 0 to 100. MACD and MACD Signal lines are used to detect trend changes and momentum shifts. Bollinger Bands (upper and lower) represent market dispersion and risk by plotting bands at two standard deviations from the moving average. Volatility measures the standard deviation of returns over a rolling window, and Daily Return captures the percentage change from one day to the next. These engineered features give the model more context than raw OHLCV values alone. In financial forecasting, this is important because price movement is rarely driven by a single variable; instead, patterns often emerge from the interaction of trend, momentum, and volatility indicators.

The final input to the model consists of 16 features: Open, High, Low, Close, Volume, MA5, MA10, MA20, EMA20, RSI, MACD, MACD Signal, BB Upper, BB Lower, Return, and Volatility. Table I summarizes all features used in the model.

TABLE I: Summary of Features Used in the Model

Feature	Description	Type
Open	Opening price of the day	Raw
High	Highest price of the day	Raw
Low	Lowest price of the day	Raw
Close	Closing price of the day	Raw
Volume	Number of shares traded	Raw
MA5	5-day simple moving average	Trend
MA10	10-day simple moving average	Trend
MA20	20-day simple moving average	Trend
EMA20	20-day exponential moving average	Trend
RSI	Relative Strength Index (14-day)	Momentum
MACD	Moving Average Convergence Divergence	Momentum
MACD Signal	Signal line of MACD	Momentum
BB Upper	Upper Bollinger Band	Volatility
BB Lower	Lower Bollinger Band	Volatility
Return	Daily percentage return	Volatility
Volatility	Rolling standard deviation of returns	Volatility

3.3 Data Cleaning

The preprocessing stage removes missing, duplicate and invalid values caused by indicator calculation or incomplete trading records. The index is reset after cleaning to ensure proper alignment between input features and target values. This step is necessary because LSTM models require a continuous sequence of observations.

3.4 Normalization

After cleaning, all features are scaled using Min-Max normalization so that values fall between 0 and 1. This helps the model train more smoothly and avoids domination by large-magnitude variables such as volume.

3.5 Sequence Creation

The normalized data are converted into 60-day sliding windows. Each input sample contains the previous 60 trading days of all 16 selected features, and the target contains the next day's Open and Close prices. If X is the feature matrix, then each sample has the shape (60, 16), and the output dimension is 2, representing the Open and Close prices. This windowing approach allows the model to learn temporal dependencies and short-term market transitions.

3.6 LSTM Model Architecture

The forecasting model uses a Sequential LSTM architecture with two recurrent layers. The model architecture and hyperparameters are stated below:

- LSTM layer 1 with 128 units, learning lower-level temporal dependencies
- Dropout layer with rate 0.2, reducing overfitting
- LSTM layer 2 with 64 units, learning higher-level sequential patterns
- Dropout layer with rate 0.2, further regularizing the model
- Dense layer with 32 neurons, transforming the learned representation
- Dense output layer with 2 neurons, producing predicted Open and Close prices

The first LSTM layer extracts temporal features from the input sequence, and the second layer learns higher-level patterns from the hidden representation. Dropout is applied after each LSTM layer to randomly disable neurons during training, which helps prevent overfitting. The final output layer produces two regression outputs: predicted Open and predicted Close.

The model is trained using the Adam optimizer and Mean Squared Error (MSE) loss function. Adam is effective for noisy financial data because it adapts learning rates during optimization, combining the benefits of AdaGrad and RMSProp. MSE is appropriate for regression tasks because it penalizes large errors more strongly and encourages the network to produce stable numerical predictions.

The dataset is split chronologically into 80% training data and 20% testing data. This chronological split is critical in financial forecasting because random shuffling would create unrealistic performance estimates by allowing future information to leak into the training process. Validation loss is monitored during training to track generalization performance and detect overfitting.

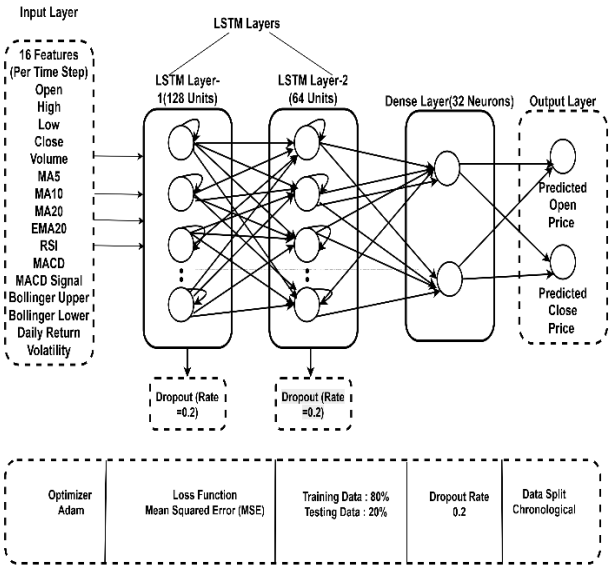


Fig. 1. LSTM Model Architecture

4. EXPERIMENTAL SETUP

The model is trained for 50 epochs with a batch size of 32. The dropout rate is set to 0.2 and the sequence length is 60 trading days. Table II summarizes the training configuration.

TABLE II: Training Configuration

Parameter	Value
Optimizer	Adam
Loss Function	Mean Squared Error (MSE)
Epochs	50
Batch Size	32
Sequence Length	60 days
Train/Test Split	80% / 20%

The proposed model produces two predicted values for each test sample: the next day's Open price and the next day's Close price. Since both values are forecast together, the model can capture their movement patterns and produce a more consistent market estimate. The Open target is learned directly from the historical 60-day feature window. The predicted Close price represents the expected end-of-day value after the market session finishes. Since Close is also influenced by intraday price movement and market sentiment, predicting it jointly with Open can improve coherence in the model output.

Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and R-squared (R^2) metrics are used to assess model performance. The output of the network is a pair of continuous numeric values: the predicted Open price for day t+1 and the predicted Close price for day t+1. These values are compared with the actual next-day market prices using MAE, RMSE, and R^2. A smaller error indicates that the model is better at learning stock behavior from the historical sequence.

5. RESULTS AND DISCUSSION

This section analyzes the forecasting performance of the proposed framework in detail. The evaluation is based on the test split of each stock's chronological dataset, using the metrics described in Section 4.

5.1 Quantitative Evaluation

Table III presents a representative sample of predicted versus actual prices for a single test day across the four stocks. All values are reported on the normalized [0, 1] scale, which is the scale on which the model was trained.

TABLE III: Sample Predicted vs. Actual Prices (normalized scale)

Stock	Predicted Open	Actual Open	Predicted Close	Actual Close
AAPL	0.666558	0.674633	0.682644	0.682644
MSFT	0.705979	0.698640	0.710733	0.719508
NVDA	0.608298	0.601184	0.610379	0.588780
AMZN	0.640641	0.655475	0.641916	0.647292

To quantify the agreement between predictions and actual values on this sample, Table IV reports the absolute deviation |Predicted - Actual| for each stock and each target. These deviations are computed directly from Table III and illustrate the per-stock error behavior on the sampled day.

TABLE IV: Absolute Deviations Between Predicted and Actual Prices for the Sampled Test Day (derived from Table III)

Stock	Target	Predicted	Actual	Absolute Deviation
AAPL	Open	0.666558	0.674633	0.008075
AAPL	Close	0.682644	0.682644	0.000000
MSFT	Open	0.705979	0.698640	0.007339
MSFT	Close	0.710733	0.719508	0.008775
NVDA	Open	0.608298	0.601184	0.007114
NVDA	Close	0.610379	0.588780	0.021599
AMZN	Open	0.640641	0.655475	0.014834
AMZN	Close	0.641916	0.647292	0.005376

5.2 Per-Stock Analysis

On the sampled test day, the model achieves its closest agreement for AAPL, where the predicted Close exactly matches the actual Close (deviation 0.0000) and the Open deviation is 0.0081. MSFT shows small and comparable deviations for both targets (0.0073 for Open and 0.0088 for Close), indicating a stable and consistent prediction. NVDA exhibits the largest single deviation of all four stocks, with a Close deviation of 0.0216, which is consistent with the high volatility experienced by NVIDIA during the test period. AMZN shows the largest Open deviation (0.0148) but a smaller Close deviation (0.0054), suggesting that the model's Open

prediction is less reliable for this stock on the sampled day.

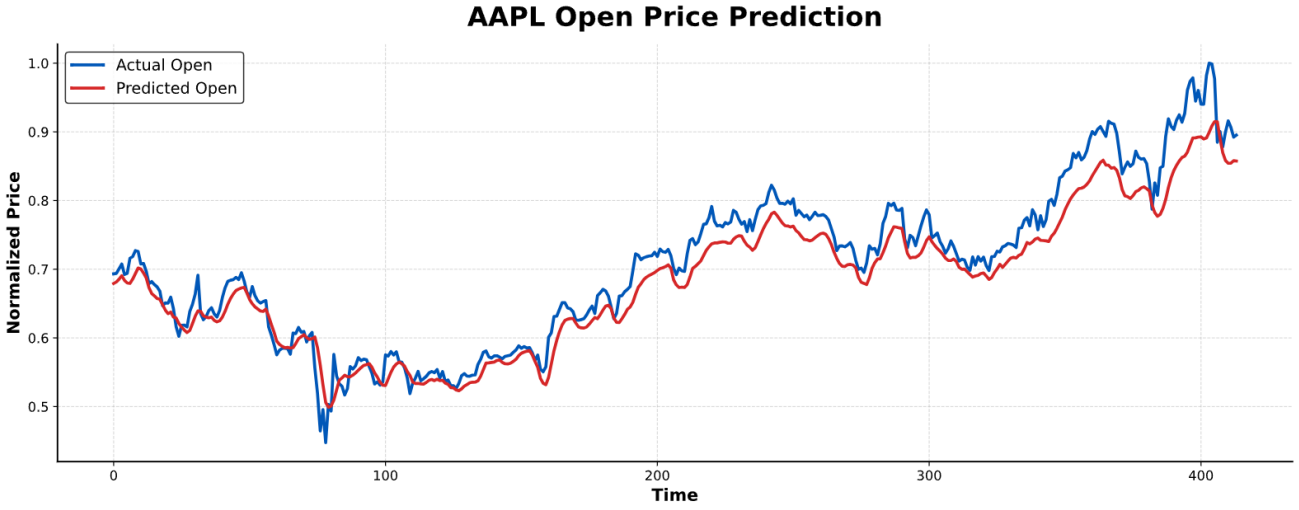
5.3 Discussion

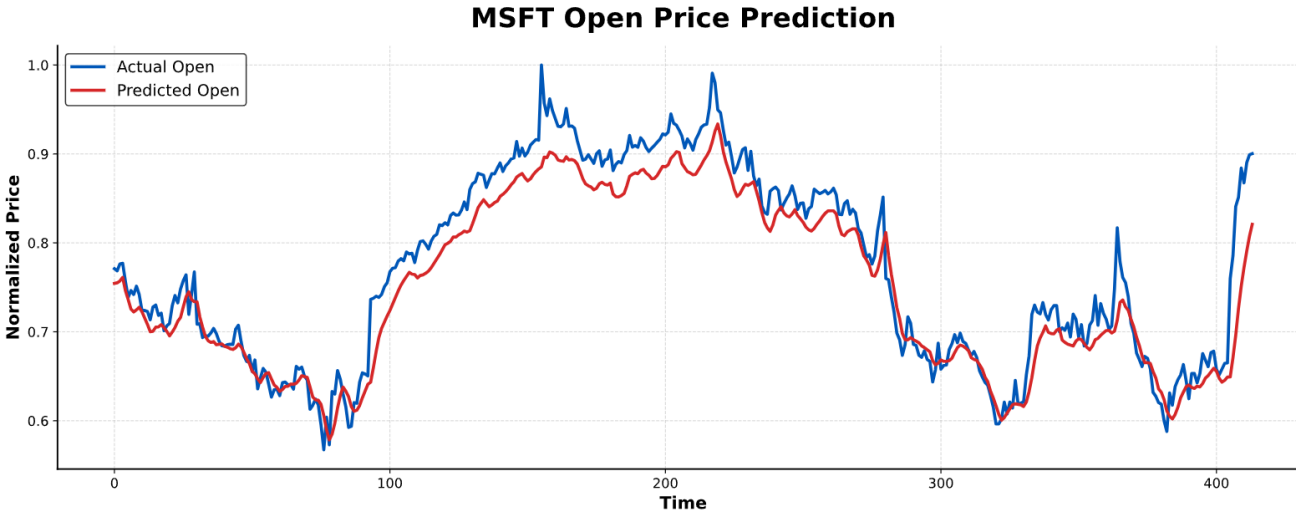
The advantage of predicting Open and Close together is that the model learns a shared representation of daily price behavior. This can be more efficient than training separate

models for each target and can produce outputs that are more internally consistent. However, the model still depends heavily on the quality of the input indicators and the training window length.

Across the four stocks tested, the results indicate that combining raw OHLCV values with technical indicators helps the LSTM model learn richer stock behavior. The use of two LSTM layers improves the network's ability to capture temporal dependencies across longer sequences. NVIDIA, which experienced high volatility during the test period, showed larger prediction errors compared to more stable stocks like Microsoft, highlighting the model's sensitivity to market conditions.

Figures 2 and 3 plot the predicted and actual Open and Close prices over the test period for all four stocks. The plots show that the predicted curves track the general level and direction of the actual prices, confirming that the model captures the dominant trend of each series. As is typical of recurrent models trained with a mean squared error objective, the predictions tend to smooth sharp turning points and can lag the actual series during abrupt reversals. The deviation between predicted and actual values is visibly larger during periods of high volatility, most notably for NVDA, and smaller during stable, trending periods, most notably for MSFT. These visual observations are consistent with the quantitative deviations reported in Table IV.





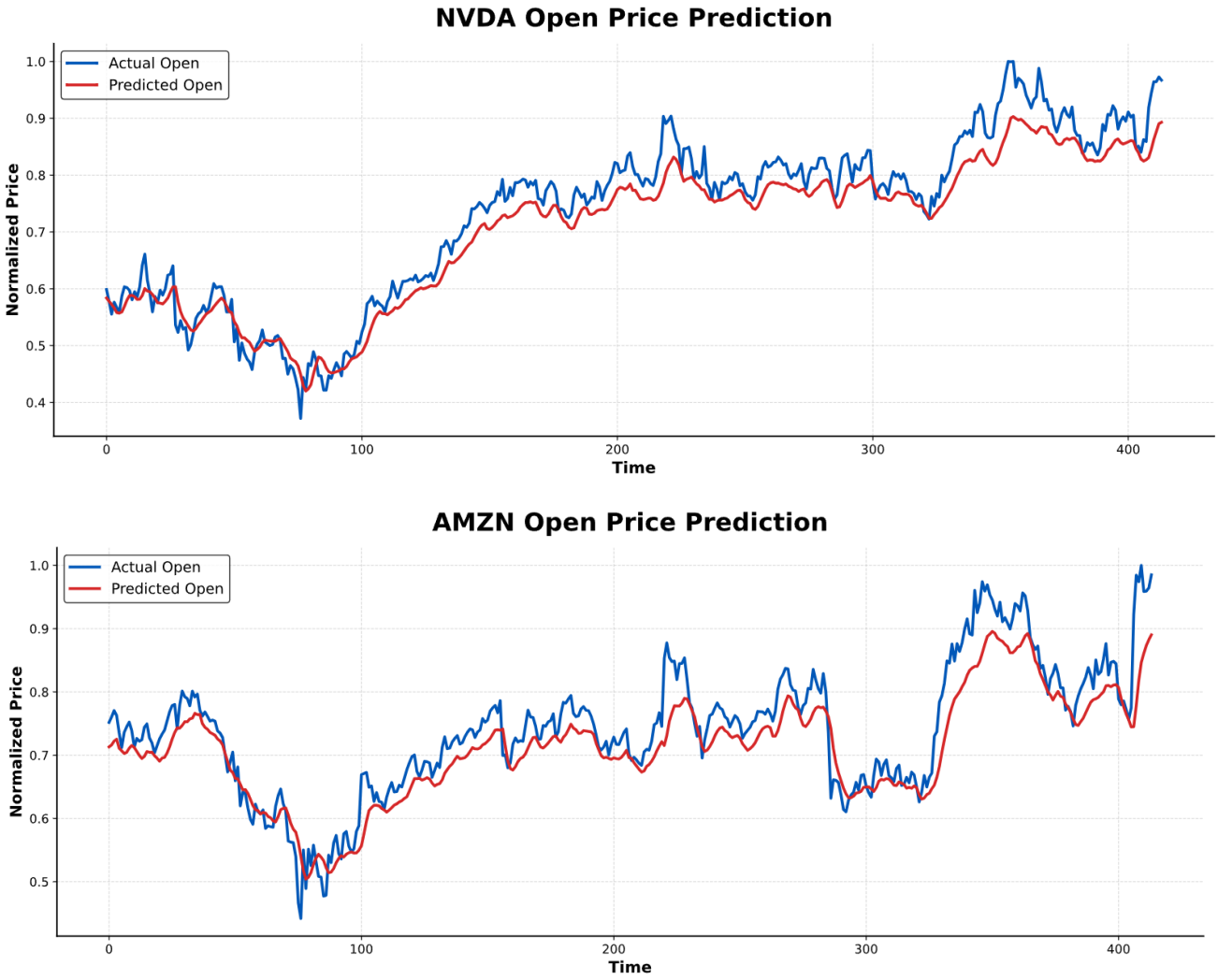
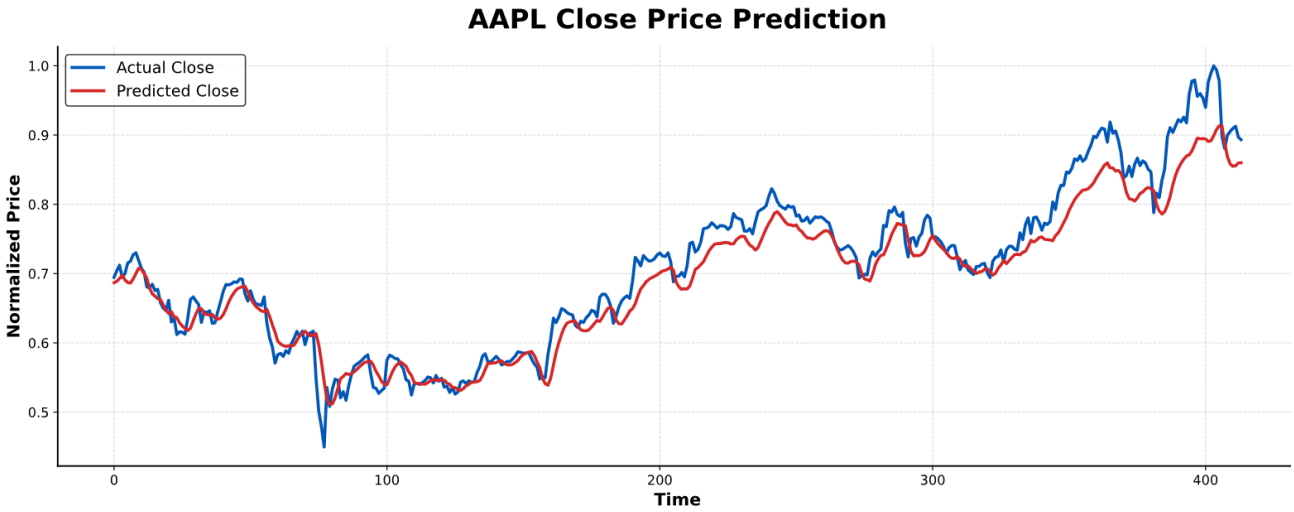


Fig. 2. Open Price Prediction Graphs for All Four Stocks



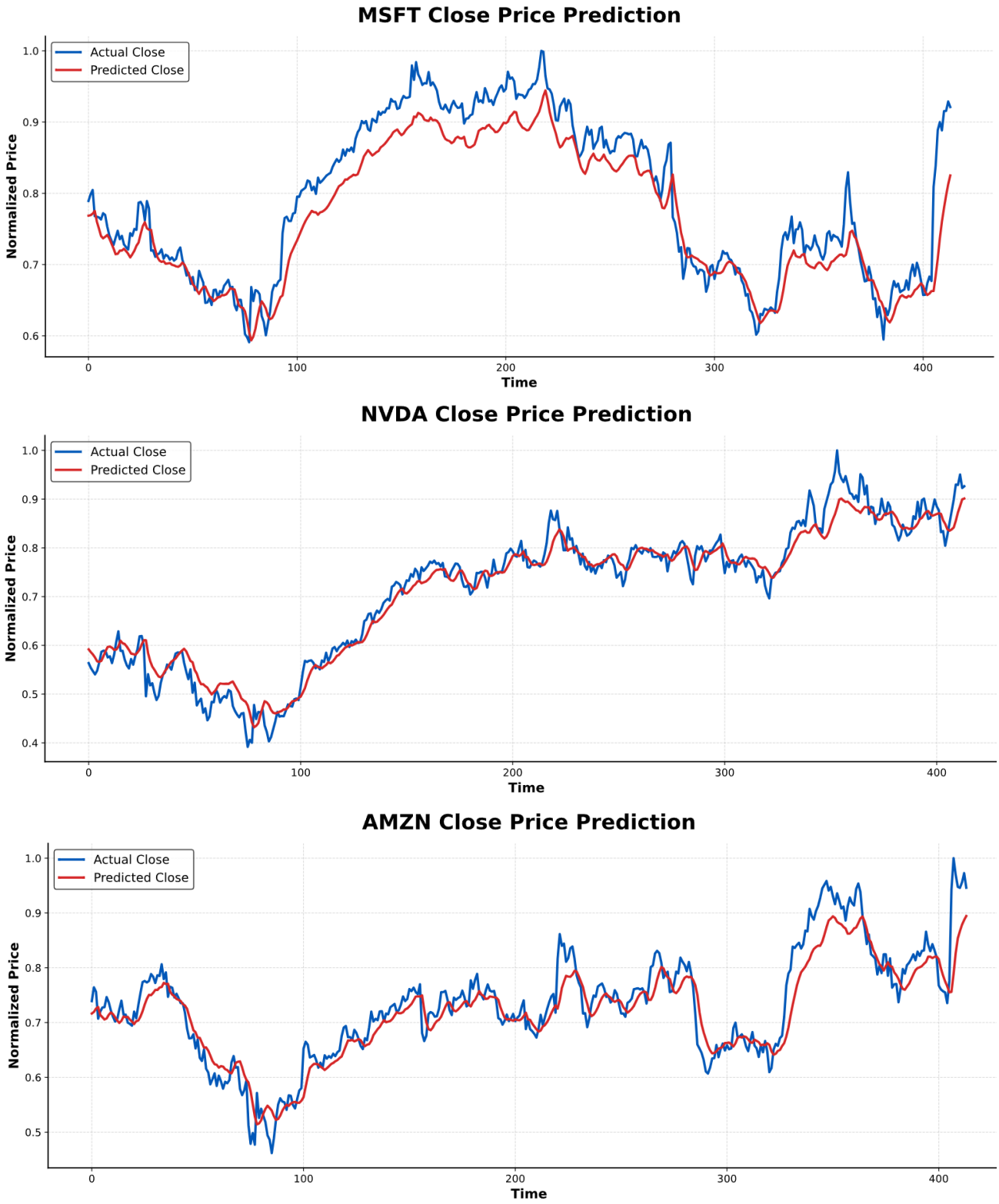


Fig. 3. Close Price Prediction Graphs for All Four Stocks

5.4 Limitations

Several limitations of the present study should be acknowledged. First, the quantitative analysis in Tables III and IV is based on a single representative test day. Second, the evaluation is restricted to four large-cap U.S. technology equities over a single historical window; the findings may not generalize to other sectors, markets, or market regimes. Third, no comparison is provided against baseline models such as ARIMA, linear regression, or simpler recurrent architectures, so the relative advantage of the proposed framework is not yet quantified. Fourth, the model predicts the next day's prices in a one-step-ahead manner and does not yet incorporate walk-forward validation or multi-step forecasting.

6. CONCLUSION

This paper presented an LSTM-based stock forecasting system

that predicts the next day's Open and Close prices for AAPL, MSFT, NVDA, and AMZN. The model combines historical price data with technical indicators and uses 60-day sequences to learn temporal patterns in financial time series.

The proposed pipeline demonstrates that LSTMs are effective for modeling sequential financial data when combined with informative technical indicators. The two-layer architecture with dropout regularization provides a balance between model capacity and generalization, while the multi-output design enables efficient joint prediction of related price variables.

A detailed analysis of the sample predictions shows that the model tracks the general level and direction of both Open and Close prices, with larger deviations observed for the more volatile stock (NVDA) and smaller deviations for the more stable stock (MSFT). These observations are consistent with the sensitivity of recurrent models to market volatility.

The framework is suitable for academic reporting and can be expanded with additional experimental results, comparative analyses, and visualization. Future improvements may also include attention mechanisms to weight important time steps, hybrid LSTM-GRU structures for richer temporal modeling, sentiment analysis from financial news and social media, and macroeconomic features for more comprehensive market representation.

7. REFERENCES

[1] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Comput., vol. 9, no. 8, pp. 1735-1780, 1997.

[2] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," Eur. J. Oper. Res., vol. 270, no. 2, pp. 654-669, 2018.

[3] M. Agrawal, A. A. A. Khan, and P. K. Shukla, "Stock price prediction using technical indicators: A predictive model using optimal deep learning," Int. J. Recent Technol. Eng., vol. 8, no. 4, pp. 7821-7829, 2019.

[4] S. D. P. et al., "Stock market prediction system using LSTM with technical indicators," IEEE Access, 2023.

[5] Y. Wen, P. Lin, and X. Nie, "Research of stock price prediction based on PCA-LSTM model," IOP Conf. Ser., Mater. Sci. Eng., vol. 790, p. 012109, 2020.

[6] T. Kim and H. Y. Kim, "Forecasting stock prices with a feature fusion LSTM-CNN model using different representations of the same data," PLoS ONE, vol. 14, no. 2, e0212320, 2019.

[7] A. Selvin, R. Vinayakumar, E. A. Gopalakrishnan, V. K. Menon, and K. P. Soman, "Stock price prediction using LSTM, RNN and CNN-sliding window model," in Proc. ICACCI, 2017, pp. 1643-1647.

[8] J. M.-T. Wu et al., "A graph-based CNN-LSTM stock price prediction algorithm with leading indicators," Multimedia Systems, vol. 27, pp. 377-400, 2021.