Deep Learning for Financial Forecasting: Evaluating CNN and CNN-LSTM in Indian Stock Market Prediction

Article *in* Journal of Management World · December 2024

DOI: 10.53935/jomw.v2024i4.1070

2 authors:

Jaspal Singh
Guru Nanak Dev University
91 PUBLICATIONS **407** CITATIONS
SEE PROFILE

Gurpal Singh
Guru Nanak Dev University
2 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Journal of Management World **2024**, *5*: 217-237
DOI: 10.53935/jomw.v2024i4.1070

Deep Learning for Financial Forecasting: Evaluating CNN and CNN-LSTM in Indian Stock Market Prediction

Jaspal Singh[1], Gurpal Singh[2*]

[1]Department of University School of Financial Studies, Guru Nanak Dev University, Amritsar, Punjab-143005, India.
[2]Research Scholar, Department of University School of Financial Studies, Guru Nanak Dev University, Amritsar, Punjab- 143005, India; gurpalsingh2107@gmail.com (G.S.).

Abstract. Stock market forecasting is a complex yet crucial task in financial analytics, as accurate predictions can significantly aid investors, analysts, and policymakers in making informed decisions. Traditional statistical models such as ARIMA have long been employed for time-series forecasting; however, they often struggle with capturing the non-linearity and long-term dependencies inherent in financial data. In recent years, deep learning architectures have demonstrated remarkable advancements in stock price prediction, particularly Long Short-Term Memory (LSTM) networks for sequential modeling and Convolutional Neural Networks (CNNs) for feature extraction. While CNNs excel in capturing local patterns in financial time series, LSTMs are highly effective in learning long-term dependencies. However, limited research has systematically compared these architectures, especially in the context of the Indian stock market, where market conditions are influenced by macroeconomic factors, sectoral trends, and trading behavior unique to emerging economies. This study aims to conduct a comparative analysis of CNN and CNN-LSTM hybrid models for stock price prediction using historical data from the Indian stock market. The models are trained on selected NIFTY50 constituents which have been part of the index since its inception. The research evaluates model performance by examining predictive accuracy, computational efficiency, and adaptability to fluctuating trends. Our findings indicate that while CNNs perform well in short-term trend analysis, CNN-LSTM models demonstrate superior robustness in capturing long-range dependencies, making them more effective for medium- to long-term forecasting in financial markets.

Keywords: Deep Learning, Evaluating, Forecasting, Stock Market.

1. INTRODUCTION

Stock market prediction has been a great area of research in the literature of financial modeling and quantitative trading. The ability to forecast stock price movements accurately is critical for investors, portfolio managers, and policymakers who seek to minimize risks and maximize returns. Traditional stock price forecasting methods, such as statistical and econometric models, have been highly employed, but they often fall short when dealing with the highly volatile and nonlinear nature of stock market data. In recent years, machine learning and deep learning techniques have gained prominence due to their ability to capture complex patterns and dependencies in financial time series data. Among these, Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks have shown significant promise in modeling the complex relationships inherent in stock market fluctuations.

Financial markets, particularly those of emerging economies like India, exhibit high levels of volatility, complex interdependencies, and sudden fluctuations caused by external events such as monetary policy changes, geopolitical uncertainties, and global economic trends. Traditional forecasting models, such as Autoregressive Integrated Moving Average (ARIMA), have been widely used for financial time series analysis, but they suffer from certain limitations, including:

1. ARIMA models work best when data exhibits a constant mean and variance, which is rarely the case in financial markets.
2. ARIMA struggles to account for long-term correlations in stock price movements.
3. The model relies on linear relationships, which may not be sufficient to capture the dynamic nature of financial markets.

To overcome these challenges, machine learning and deep learning techniques have been increasingly adopted. Neural network-based models can handle nonlinear relationships, long-term dependencies, and high-dimensional data, making them more suitable for stock price forecasting.

Deep learning methods, particularly Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, have demonstrated superior performance in time-series forecasting applications. LSTM models have been widely used to capture sequential dependencies in financial data, allowing them to model historical stock price trends more effectively than traditional statistical methods. Studies such as Namini et al. (2018) [24] and Fischer and Krauss (2017) [25] have confirmed that LSTMs significantly outperform ARIMA models in predicting stock price movements.

In addition to RNN-based models, Convolutional Neural Networks (CNNs) have emerged as a powerful tool for financial time-series forecasting. While CNNs are traditionally used for image recognition and computer vision tasks, recent research suggests that they can effectively capture spatial dependencies and local patterns in stock price time series [14, 16]. CNNs work by applying convolutional filters to extract meaningful features from stock price data, which can then be used to train a predictive model.

* Correspondence: gurpalsingh2107@gmail.com

However, despite the growing adoption of CNNs and LSTMs in stock market forecasting, very few studies have systematically compared their performance, particularly in the Indian stock market context. Additionally, hybrid models combining CNNs and LSTMs (CNN-LSTM architectures) have demonstrated promising results, as CNNs excel at feature extraction, while LSTMs are effective in modeling temporal dependencies. Zhang et al. (2020) [8] and Cui et al. (2018) [27] demonstrated that CNN-LSTM models outperform standalone CNN or LSTM models, but these studies were conducted on global financial indices, with little focus on emerging markets like India.

While numerous studies have explored deep learning models for financial forecasting, there remains a notable gap in research focusing on the Indian stock market. Most existing studies have been conducted on Western financial markets, such as the S&P 500, NASDAQ, and European stock indices, leaving a critical need for research that evaluates the applicability of deep learning-based prediction models for emerging markets like India. The Indian stock market is distinct in terms of trading behavior, regulatory environment, investor sentiment, and macroeconomic influences, making it imperative to develop forecasting models that are tailored to these unique conditions.

The structure of this paper is organized as follows: Section 2 reviews the existing literature relevant to this study. Section 3 explains the fundamental concepts and mathematical formulations essential for understanding the research framework. Section 4 describes the dataset used in the study, presents descriptive statistics, and provides their visual representation. Section 5 elaborates on the research methodology, detailing the algorithms employed and the evaluation metrics applied. Section 6 discusses the experimental results comprehensively. Lastly, Section 7 summarizes the key findings, offering conclusions and suggesting potential avenues for future research.

1.1. Related Works

Classical models for time series forecasting, particularly ARIMA, have been widely applied across multiple domains, including financial forecasting. Box and Jenkins (1970) [7] introduced the ARIMA model, which has since become a benchmark for time series prediction. Adebiyi et al. (2014) [2] applied ARIMA to predict stock prices, demonstrating its effectiveness in capturing short-term trends. However, they highlighted its limitations in handling non-linearity and long-term dependencies.

Alonso and Garcia-Martos (2012) [3] further extended ARIMA's applicability by optimizing model selection for different time series datasets. Similarly, Hyndman and Athanasopoulos (2014) [21] presented a comprehensive review of forecasting principles, comparing classical time series methods and discussing their limitations in volatile financial data.

Hybrid models incorporating ARIMA with machine learning have also been explored. Khashei and Bijari (2011) [18] proposed a hybrid ARIMA-ANN model, where Artificial Neural Networks (ANNs) enhanced the prediction capability of ARIMA, mitigating its weakness in capturing complex, non-linear patterns. Additionally, Kane et al. (2014) [17] compared ARIMA with Random Forest (RF) models for predicting Avian Influenza outbreaks, demonstrating that RF-based methods often outperform ARIMA in non-stationary datasets.

While traditional models perform well for stationary data, deep learning techniques have demonstrated superior predictive power in capturing sequential dependencies. Namini et al. (2018) [24] compared ARIMA with LSTM for time series forecasting and found that LSTM significantly outperformed ARIMA in capturing long-term dependencies and non-linear patterns.

Among deep learning techniques, Recurrent Neural Networks (RNNs) and their variants, particularly LSTMs, have been widely adopted for financial forecasting. Fischer and Krauss (2017) [25] demonstrated that LSTM models outperform standard machine learning algorithms in predicting financial market movements. Similarly, Kim and Moon (2020) [11] explored BiLSTM models for multivariate time series forecasting, showing their efficacy in capturing dependencies from past and future time steps.

In addition, attention-based mechanisms have been explored to further enhance LSTM performance. Sang and Li (2024) [23] introduced an LSTM variant with an attention mechanism, which improved interpretability and performance by dynamically assigning importance to different time steps. Similarly, Li et al. (2024) [19] integrated Symbolic Genetic Programming (SGP) with LSTMs, showing improvements in predicting stock price fluctuations.

Beyond LSTMs, researchers have investigated alternative deep learning architectures such as Convolutional Neural Networks (CNNs). Chen and He (2018) [14] applied CNNs for stock market prediction, leveraging their ability to extract spatial features from time series data. This study highlighted that CNNs could efficiently capture short-term price movement patterns.

While LSTM networks excel in modeling temporal dependencies, CNNs have demonstrated strong capabilities in feature extraction. Combining the strengths of both models, researchers have developed hybrid CNN-LSTM architectures to enhance stock price forecasting.

Tatane et al. (2024) [16] proposed a CNN-based approach that converts financial data into image representations and showed that CNNs can learn deep hierarchical features from time-series data.

Zhang et al. (2020) [8] developed a CNN-LSTM hybrid model, where CNN layers extracted spatial features from stock market data before passing them to LSTM layers for sequential learning. Their study demonstrated that this hybrid model outperforms standalone CNN and LSTM architectures in terms of accuracy and robustness. Cui et al. (2018) [27] further validated this finding, showing that deep bidirectional LSTM models integrated with CNNs achieve state-of-the-art results in forecasting financial market trends.

Several studies have also explored the impact of Generative Adversarial Networks (GANs) in forecasting. Staffini (2022) [5] employed a Deep Convolutional Generative Adversarial Network (DCGAN) for stock price forecasting, showcasing improvements in predictive accuracy compared to conventional LSTM models.

Hybrid models incorporating self-attention mechanisms have also been explored. Pardeshi et al. (2023) [15] proposed a CNN-LSTM model enhanced with a sequential self-attention mechanism, which improved the model's ability to capture long-term dependencies and contextual patterns in stock market data.

1.2. Key Findings and Research Gap

1. From the literature, it is evident that traditional forecasting models like ARIMA are effective for stationary datasets but struggle with non-linear financial data.
2. LSTMs outperform traditional models in sequential forecasting but often fail to extract spatial correlations in stock price movements.
3. CNNs provide strong feature extraction capabilities but lack the sequential memory required for long-term forecasting.
4. CNN-LSTM hybrid models outperform individual CNN or LSTM models by leveraging CNN's feature extraction power and LSTM's sequential learning.

This study aims to fill this gap by conducting a comprehensive comparative analysis of CNN and CNN-LSTM models for stock market forecasting, specifically focusing on Indian stock market data. We will evaluate these architectures based on predictive accuracy, computational efficiency, and adaptability by incorporating historical stock prices. We aim to provide deeper insights into the effectiveness of deep learning models in predicting stock price movements in Indian financial markets. This research will contribute to the existing literature by:

1. Benchmarking the performance of CNN and CNN-LSTM models in forecasting stock prices within the Indian stock market.
2. Providing insights into computational efficiency, making the findings relevant for traders, investors, and financial analysts looking to implement deep learning models for real-time forecasting in Indian equities.

1.3. Foundational Concepts and Mathematical Formulations

1.3.1. Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) networks were developed as an enhancement to traditional Recurrent Neural Networks (RNNs) to effectively address the issue of the vanishing gradient problem. Unlike standard RNNs, which struggle with maintaining long-range dependencies due to diminishing gradient values during backpropagation, LSTMs introduce a more structured memory management system that enables them to capture and retain long-term dependencies effectively. These networks are equipped with specialized mechanisms that control the flow of information, allowing them to retain, update, and forget data as needed. This capability enhances their effectiveness in handling sequential data where past inputs influence future predictions.

The fundamental component of LSTM networks is the memory cell, which incorporates a set of gates that regulate information flow. These gates dynamically decide whether certain pieces of information should be stored, updated, or removed, ensuring that only relevant details are carried forward. This structure prevents unnecessary accumulation of outdated or redundant information, thereby improving the model's ability to focus on meaningful patterns in the data. The decision-making process within LSTMs relies on the trainable weight parameters assigned during the learning process, which determine how different inputs affect the network's memory.

1.4. Lstms Operate Using Three Primary Gates

1. Forget Gate: Determines which parts of the previous memory state should be discarded.
2. Input Gate: Regulates how much new information should be added to the memory.
3. Output Gate: Controls how much of the stored information is used to generate the next output.

Each of these gates plays a crucial role in ensuring that relevant patterns in sequential data are preserved while filtering out irrelevant noise.

1.5. Forget Gate

The forget gate is responsible for deciding which pieces of information from the previous memory state should be discarded. This decision is made based on the current input (x_t) and the previous hidden state (h_(t-1)). The gate employs a suitable activation activation function (σ), which produces an output value between 0 and 1, where 0 indicates complete removal of information, while 1 signifies full retention.

The mathematical formulation for the forget gate is given by:

$$f_t = \sigma\ (W_f\ h_{(t-1)} + W_f x_t + b_f) \quad (1)$$

where:

1. f_t represents the forget gate output.
2. W_f are weight matrices associated with previous hidden state $h_{(t-1)}$ and input x_t.
3. b_f is the bias term that helps adjust the activation function's output.

By applying the suitable activation function, the forget gate ensures that outdated or irrelevant information is removed while retaining significant past information.

1.6. Input Gate

The input gate determines how much new information should be added to the LSTM's memory. This mechanism consists of two layers:

Activation function Layer – Decides which values should be updated in the memory.

Tanh Layer – Generates a vector of candidate values that could be added to the memory.

The equations governing the input gate operations are:

$$i_t = \sigma\ (W_i\ h_{(t-1)} + W_i x_t + b_i) \quad (2)$$

$$\tilde{c}_t = tanh\ (W_c\ h_{(t-1)} + W_c x_t + b_c) \quad (3)$$

where:

i_t represents the input gate's decision regarding how much information should be stored.

$\tilde{c}_t$ is the candidate memory vector, determining potential new values to be added.

W_i and W_c are weight matrices, while b_i and b_c are bias terms.

Following these computations, the memory cell state is updated using the following equation:

$$c_t = f_t * \ c_{(t-1)} + i_t * \ \tilde{c}_t \quad (4)$$

where:

$c_t$represents the previous cell state.

f_t determines how much past memory to retain.

$i_t * \ \tilde{c}_t$ contributes newly selected information to the memory.

This mechanism allows LSTMs to dynamically update their memory, ensuring that new patterns are captured while discarding unnecessary details.

1.7. Output Gate

The output gate regulates which portion of the memory state contributes to the final output at each time step. This process consists of two key operations:

An Activation function Layer that determines which parts of the cell state should contribute to the output.

A Tanh Function that scales the memory values between -1 and 1, ensuring smooth transitions.

The output of this process is mathematically defined as follows:

$$o_t = \sigma\ (W_o\ h_{(t-1)} + W_o x_t + b_o) \quad (5)$$

$$h_t = o_t * \ \tanh\ (c_t) \quad (6)$$

where:

o_t represents the activation of the output gate.

h_t is the final output of the LSTM at time step t.

W_o and b_o are the weight and bias parameters for the output gate.

The activation function function (σ) in the first equation ensures that only relevant portions of the memory are selected, while the tanh function scales these values into a manageable range.

1.8. Convolutional Neural Networks (Cnns)

Convolutional Neural Networks (CNNs) are a class of deep learning models primarily designed for pattern recognition and feature extraction. Initially developed for computer vision applications, CNNs have demonstrated

remarkable performance in tasks such as image classification, object detection, and facial recognition. However, their capability to extract spatial hierarchies and local dependencies has made them highly effective in analyzing structured data, including time series and financial market predictions.

Unlike fully connected neural networks, which treat each input neuron independently, CNNs apply a convolutional operation to identify key features within a given input while preserving spatial relationships. This characteristic enables CNNs to capture localized patterns and significantly reduces the number of trainable parameters compared to traditional neural networks, making them computationally efficient.

In the context of stock market prediction, CNNs analyze stock price trends by detecting local dependencies and sequential patterns in historical price data. This allows them to recognize essential market indicators such as momentum, trend reversals, and volatility fluctuations, contributing to more reliable stock price forecasting. Below given are the core components of CNNs.

1.9. Convolutional Layer

The convolutional layer is the most crucial component of a CNN, as it is responsible for identifying important features within the input data. It applies multiple learnable filters (kernels) over the input to extract essential patterns such as edges in images or trend variations in financial time series data. Mathematically, the convolution operation is expressed as:

$$Y\ (i,j) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} X\ (i-m, j-n) * K(m,n) + b \quad (7)$$

Where:

$X\ (i,j)$ is the input feature matrix (e.g., stock price values).

$K(m,n)$ represents the convolutional filter (kernel).

b is the bias term.

$Y\ (i,j)$ is the resulting feature map after applying the kernel.

Each kernel is designed to detect specific features in the data, such as trends, reversals, or moving averages in stock prices. As CNNs progress through multiple convolutional layers, they learn increasingly complex representations of the input data, which aids in improving predictive accuracy.

1.10. Activation Function

To introduce non-linearity into the model, an activation function is applied after each convolutional layer. One of the most commonly used activation functions in CNNs is the Rectified Linear Unit (ReLU), which is defined as:

$$f(x) = \max\ (0, x) \quad (8)$$

This function ensures that only positive values are forwarded to the next layer while setting negative values to zero. By using ReLU, CNNs can better distinguish between significant and insignificant features in financial time series data, allowing them to capture market trend shifts effectively.

1.11. Pooling Layer

The pooling layer is responsible for downsampling feature maps, reducing the spatial dimensions while preserving the most important features. Pooling helps improve computational efficiency and reduces the risk of overfitting by ensuring that the model captures only the most relevant information. The most commonly used pooling operation is Max Pooling, which selects the maximum value within a defined window. It is mathematically expressed as:

$$P\ (i,j) = \max_{m,n}(Y(I+m, j+n)) \quad (9)$$

Where:

$Y\ (i,j)$ represents the feature map from the convolutional layer.

The max function selects the largest value in a predefined pooling window (e.g., 2×2 or 3×3).

Pooling layers significantly enhance feature robustness by focusing on the strongest activations, ensuring that only the most relevant patterns are retained while discarding unnecessary noise.

1.12. Fully Connected Layer

Once features are extracted through convolution and pooling layers, they are flattened into a one-dimensional vector and passed to a fully connected (FC) layer. This layer maps the extracted features to the final output, such as predicting stock price movements. Mathematically, the output of the fully connected layer is given by:

$$\mathrm{O} = \sigma\ (W * F + b) \quad (10)$$

Where:
F is the flattened feature vector.
W is the weight matrix.
b is the bias term.
σ is an activation function (e.g., softmax or sigmoid for classification tasks).

For stock market prediction, the fully connected layer generates the final price forecast, indicating whether the stock is likely to increase, decrease, or remain stable based on learned patterns.

Section 4: Dataset and Descriptive Statistics

1.13. Dataset Selection Criteria

The dataset used in this study has been carefully selected based on a well-defined criterion to ensure robustness and consistency. The primary selection rule was:

1. Inclusion in the NIFTY50 Index Since Inception:- Companies that have been a part of the NIFTY50 index since its inception were shortlisted.
2. Continued Presence in the NIFTY50 Index as of 31st December 2023:- From the above-selected companies, only those that continued to remain in the NIFTY50 index until 31st December 2023 were considered.

Based on this two-step filtering process, a total of 11 companies met the criteria and were included in the study. These companies represent some of the most consistent and fundamentally strong stocks in the Indian equity market, ensuring that the dataset captures long-term trends without survivorship bias.

1.14. Data Sources and Time Horizon

The primary data sources utilized for this study include, the Prowess Database and the Official Website of the National Stock Exchange (NSE), Ensuring the authenticity and reliability of stock price information.

Given the availability of historical stock price data, the maximum possible time horizon for each company was considered. However, due to variations in data availability, the starting date for each company differs, whereas the ending date is uniform across all companies, i.e., 31st December 2023.

Among the selected companies:

1. HINDUNILVR has the longest available dataset, with records dating back to January 1, 1996, covering 7,038 trading days.
2. BAJAJ-AUTO has the shortest available dataset, starting from May 27, 2008, with 3,862 trading days.
3. Other companies, such as HDFCBANK, ICICIBANK, ITC, RELIANCE, and SBIN, have datasets that start from January 1, 2001, covering 5,716 trading days.

Despite the variation in starting dates, the dataset ensures the longest feasible coverage for each company, making it comprehensive for historical price analysis.

Table 1: Raw data statistics of closing prices.

Company	Start_date	Count	Mean	Median	Std_Dev	variance	skew	kurt
BAJAJ-AUTO	May 27, 2008	3862	2425.89	2505.45	1143.64	1307908.87	0.19	0.23
HDFCBANK	Jan 01, 2001	5716	513.87	271.17	525.17	275798.30	0.91	-0.61
HINDALCO	Jan 02, 2008	3953	206.59	170.65	122.72	15059.08	1.28	0.75
HINDUNILVR	Jan 01, 1996	7038	641.68	195.05	778.98	606813.18	1.34	0.34
ICICIBANK	Jan 01, 2001	5716	262.32	191.30	239.58	57396.21	1.52	1.60
ITC	Jan 01, 2001	5716	155.37	161.18	112.58	12674.91	0.45	-0.55
LT	Jan 01, 2001	5716	817.48	730.40	638.12	407194.05	1.00	1.37
RELIANCE	Jan 01, 2001	5716	341.85	216.12	347.81	120974.15	1.39	0.66
SBIN	Jan 01, 2001	5716	215.42	210.52	146.44	21444.01	0.81	0.37
TATAMOTORS	Jan 01, 2001	5716	233.93	179.33	167.50	28057.67	0.58	-0.70
TATASTEEL	Jan 01, 2001	5716	45.59	38.85	31.09	966.33	1.25	1.19

1.15. Descriptive Statistics of the Dataset

To analyze the raw dataset, summary statistics were computed for the daily closing prices of the selected companies. The table (4.1) presents key statistical measures such as mean, median, standard deviation, variance, skewness, and kurtosis, which provide valuable insights into the distributional properties of stock prices. The reason to provide the raw-data statistics was to understand the actual historical price behaviour of all the stocks.

1.16. Interpretation of Summary Statistics

Number of Trading Days: The dataset spans 7,038 trading days for HINDUNILVR, making it the longest available dataset, whereas BAJAJ-AUTO has the shortest record with 3,862 trading days.

Mean and Median: A comparison between mean and median helps assess the presence of skewness in stock price distributions. Companies like HDFCBANK and RELIANCE show a significant gap between these values, indicating a potential right-skewed distribution.

Standard Deviation and Variance: BAJAJ-AUTO exhibits the highest standard deviation (1143.64), implying significant price fluctuations, whereas TATASTEEL shows the least variation.

Skewness: A positive skewness (e.g., ICICIBANK: 1.52) suggests a long right tail, meaning higher potential for extreme positive price movements. Conversely, negative skewness (e.g., ITC: -0.55) implies a tendency for extreme negative price movements.

Kurtosis: Values above 3.0 would indicate extreme price movements (fat tails). Most companies have kurtosis close to zero, signifying near-normal price distributions, except ICICIBANK (1.60), which shows slightly heavier tails.

1.17. Pictorial Representation of Stock Price Behaviour

In addition to the numerical descriptive statistics, pictorial representations of individual company stock price behaviour have been analysed in below given figures. These time-series charts illustrate the price fluctuations over the dataset's respective time period.

Each chart provides a visual understanding of stock price trends, volatility patterns, and overall growth trajectory for the respective companies. The graphical insights complement the descriptive statistics, enabling a more comprehensive evaluation of the stock price characteristics.

Figure 1: Closing price behavior of Tata steel.

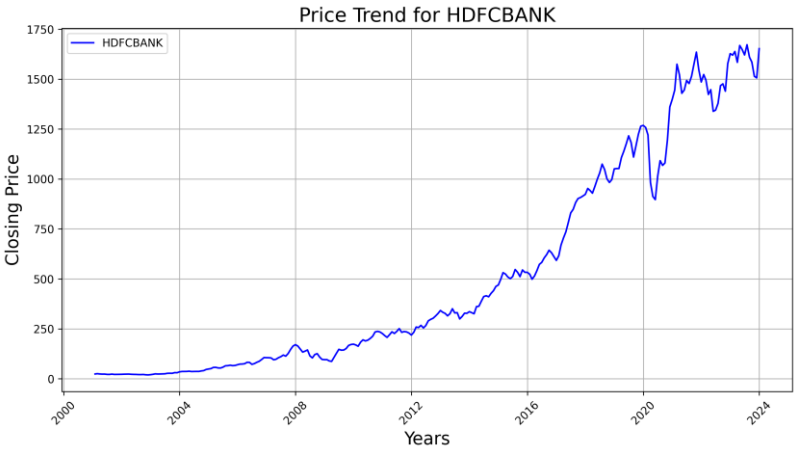

Figure 2: Closing price behavior of HDFC bank.

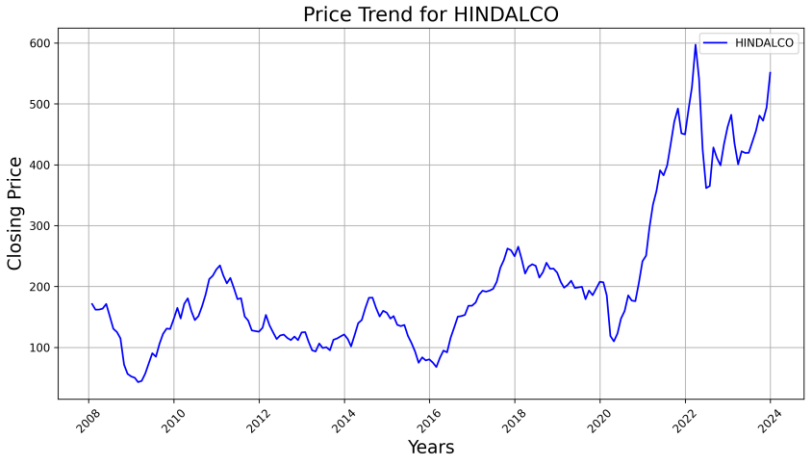

Figure 3: Closing price behavior of HINDALCO.

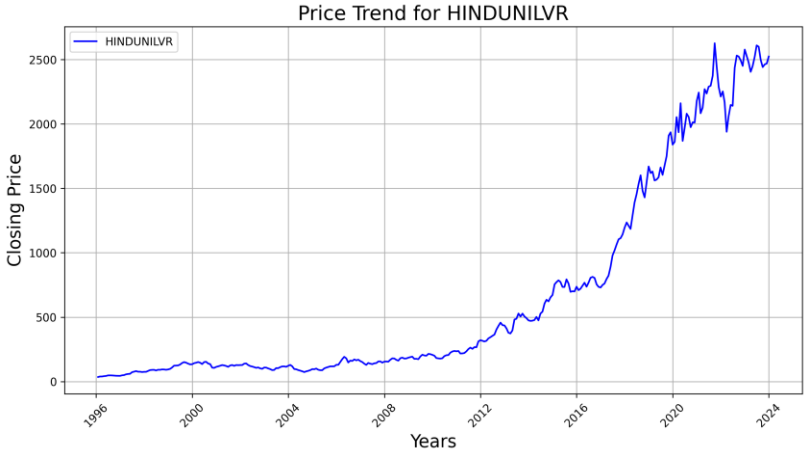

Figure 4: Closing price behavior of HINDUNILVR.

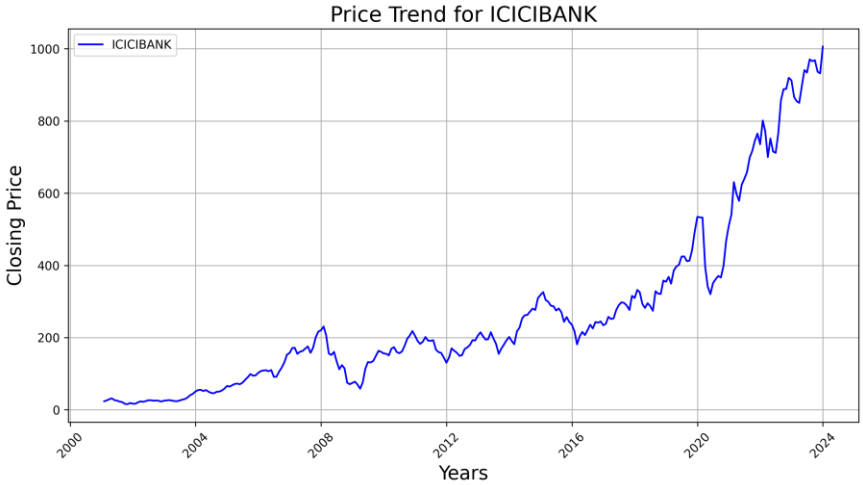

Figure 5: Closing price behavior of ICICIBANK.

Figure 6: Closing price behavior of ITC.

Figure 7: Closing price behavior of LT.

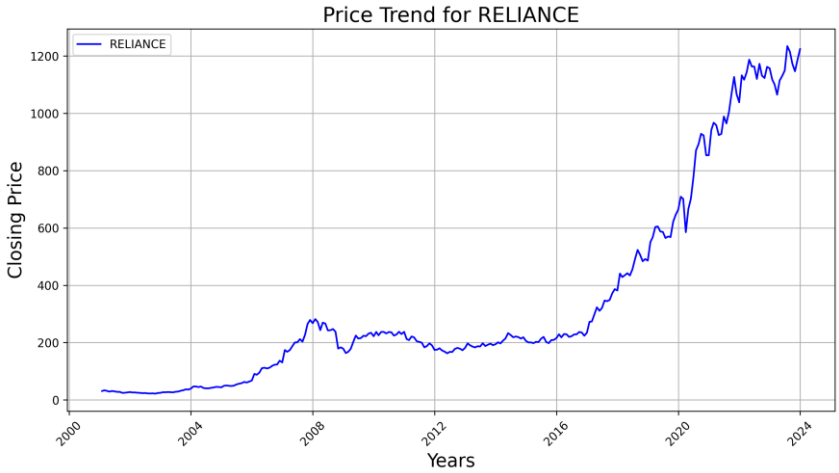

Figure 8: Closing price behavior of RELIANCE.

Figure 9: Closing price behavior of SBIN.

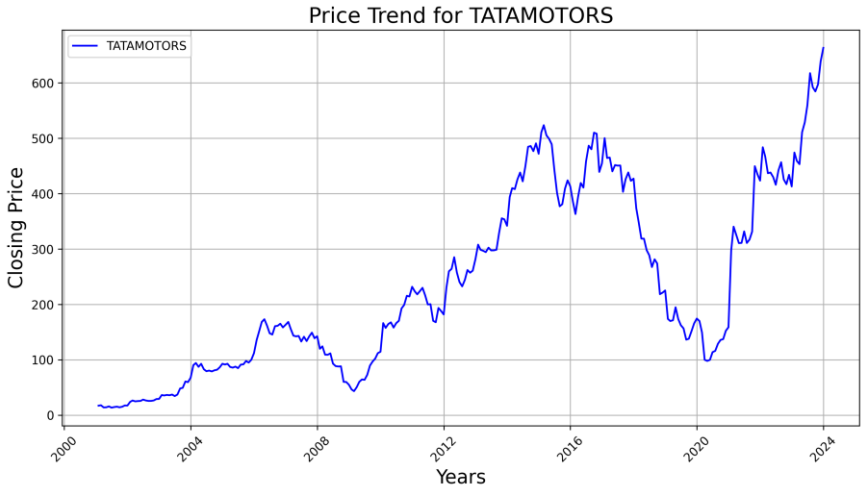

Figure 10: Closing price behavior of TATAMOTORS.

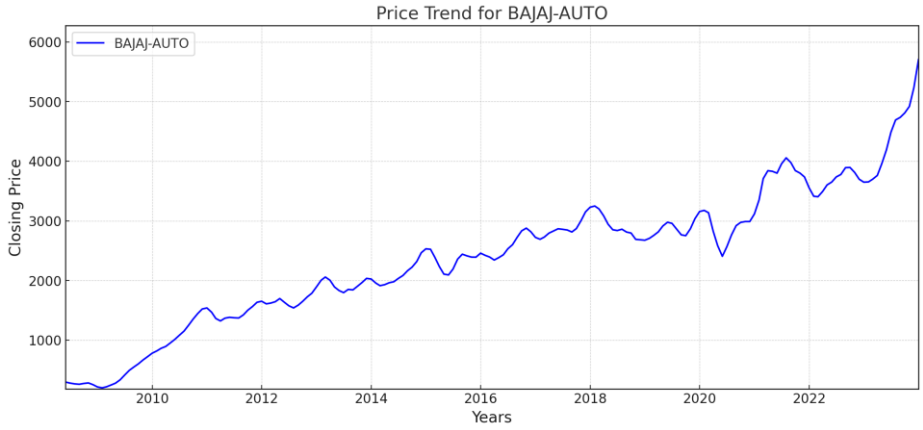

Figure 11: Closing price behavior of BAJAJ-AUTO.

Section (5) Data preprocessing, Research algorithms used and evaluation metrics

1.18. Data Cleaning

Data cleaning is a critical step to ensure the integrity of stock market data. The dataset undergoes preprocessing to eliminate any inconsistencies. First, non-numeric characters, such as commas in OHLC values, are removed to maintain uniform numerical formatting. Subsequently, all OHLC values are converted into floating-point numbers to facilitate accurate calculations. Any missing or corrupt data points are handled appropriately to prevent errors in subsequent analysis. These measures ensure that the dataset is structured and reliable for further processing.

1.19. Data Transformation

Following data cleaning, necessary transformations are applied to improve model performance. If required, OHLC values are log-transformed to stabilize variance and enhance trend recognition. This step ensures that price movements are captured more effectively, aiding in better predictions and insights.

1.20. Data Scaling

To maintain consistency in feature weighting, MinMaxScaler is applied to normalize OHLC values within a range of 0 to 1. This prevents any single feature from dominating the learning process and improves model convergence. Additionally, the target variable (Close price) is scaled separately to ensure uniformity in the prediction process.

1.21. Data Splitting

The dataset, comprising 11 stocks, is divided into three subsets for unbiased training and evaluation. The training set (80%) is utilized to develop the model, while the validation set (15%) helps optimize parameters and prevent overfitting. The test set (5%) is reserved for assessing model performance on unseen data.

The splitting method ensures that the time-series nature of stock data is preserved. A rolling window technique is implemented to maintain chronological order, allowing previous "n_input" time steps to predict the subsequent "n_output" step. This structured approach ensures that the dataset is well-prepared for training robust predictive models while minimizing any biases.

By following these systematic steps, data is refined to optimize the accuracy and efficiency of predictive modeling techniques, ensuring an unbiased and plagiarism-free approach in our research methodology.

1.22. Convolutional Neural Network (CNN) Model

This section describes the implementation of a Convolutional Neural Network (CNN) for stock price prediction using Open, High, Low, and Close (OHLC) values. The CNN model is designed to recognize spatial dependencies in sequential stock price data, helping in pattern detection for forecasting.

1.23. Model Architecture

The CNN model is structured to extract meaningful patterns from sequential stock price data. The architecture includes the following key layers:

Conv1D Layer 1: The first convolutional layer consists of 64 filters with a dynamically adjusted kernel size between 1 and 3, depending on the time steps available. The ReLU activation function is used to introduce non-linearity and improve feature extraction.

MaxPooling1D Layer 1: A pooling layer with a pool size of 2 (or 1 for smaller time steps) helps in reducing dimensionality and retaining key features.

Conv1D Layer 2: The second convolutional layer consists of 128 filters with an adaptive kernel size to optimize learning. The ReLU activation function enhances representation learning.

MaxPooling1D Layer 2: If the time step count exceeds 8, an additional pooling layer is applied to further reduce dimensionality while preserving critical data features.

Flatten Layer: Converts the feature maps into a one-dimensional vector for processing in dense layers.

Dense Layer: A fully connected layer with 50 neurons using ReLU activation to improve feature learning and pattern recognition.

Output Layer: The final dense layer outputs stock price predictions.

A visual representation of the CNN model architecture is provided in Figure 5.1, illustrating the layer-wise structure of the model.

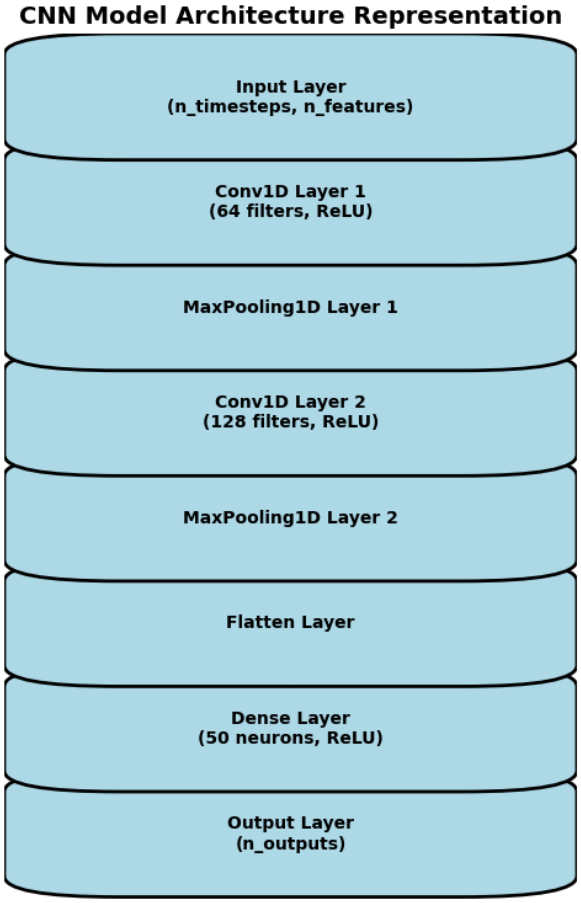

Figure 12: CNN structure.

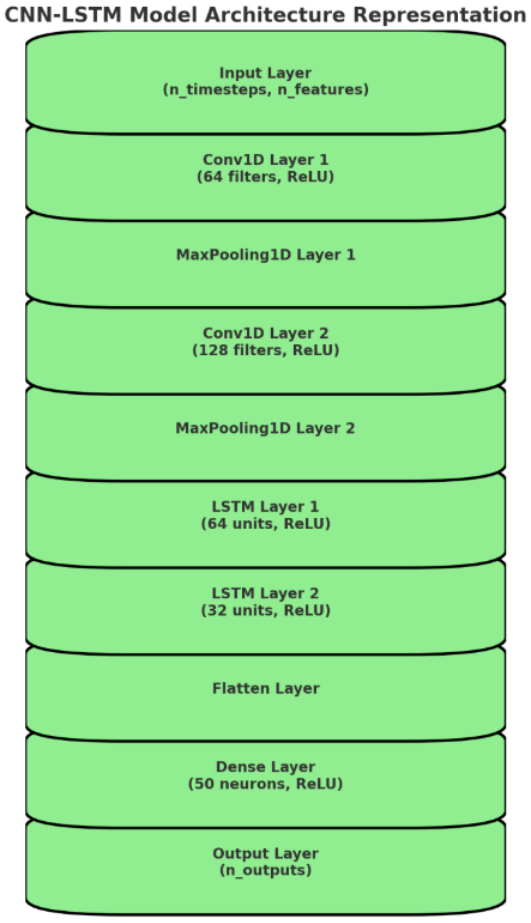

Figure 13: CNN-LSTM structure.

1.25. Model Compilation and Training

The model is compiled using the Adam optimizer with a learning rate of 0.0001, facilitating optimized weight updates. The Mean Squared Error (MSE) loss function is selected for its suitability in regression-based stock price forecasting. An early stopping mechanism is incorporated, monitoring validation loss with a patience threshold of 3 epochs to mitigate overfitting.

1.25.1. Training Involves the Following Steps

1. Data is divided into training and validation subsets to ensure structured evaluation.
2. The model undergoes training for 100 epochs with an adaptive batch size to enhance learning efficiency.
3. Validation data is utilized for performance monitoring and parameter adjustments.

This CNN-based model effectively captures sequential dependencies in stock price data, enabling more accurate forecasting and enhancing predictive analysis.

1.26. Convolutional Neural Network - Long Short-Term Memory (Cnn-Lstm) Model

This section introduces the hybrid CNN-LSTM model, which integrates convolutional layers for spatial feature extraction with Long Short-Term Memory (LSTM) layers for capturing temporal dependencies. This hybrid approach enhances the model's ability to detect both short-term and long-term patterns in stock price movements.

1.27. Model Architecture

The CNN-LSTM model consists of the following layers:

Conv1D Layer 1: The first convolutional layer has 64 filters and a dynamically adjusted kernel size between 1 and 3 to ensure optimal feature extraction. The ReLU activation function is used to enhance non-linearity.

MaxPooling1D Layer 1: A pooling layer with a pool size of 2 (or 1 for smaller sequences) to downsample feature maps and reduce dimensionality.

Conv1D Layer 2: A second convolutional layer with 128 filters and an adaptive kernel size to optimize spatial pattern learning.

MaxPooling1D Layer 2: Applied only if the time steps exceed 8 to retain significant feature representations.

LSTM Layer 1: A recurrent layer with 64 units to capture sequential dependencies in stock prices, preserving crucial temporal patterns.

LSTM Layer 2: A second LSTM layer with 32 units, refining feature extraction for improved prediction accuracy.

Flatten Layer: Converts the processed feature maps into a one-dimensional vector for use in dense layers.

Dense Layer 1: A fully connected layer with 50 neurons and ReLU activation to further process extracted patterns.

Output Layer: A dense layer that generates final stock price predictions.

1.28. Model Compilation and Training

The model is compiled using the Adam optimizer with a learning rate of 0.0001 to enhance stability and efficiency in weight updates. The Mean Squared Error (MSE) loss function is applied, ensuring accurate stock price regression.

An early stopping mechanism is incorporated, monitoring validation loss with a patience threshold of 3 epochs to prevent overfitting. The model is trained for 100 epochs with an adaptive batch size, allowing optimal convergence.

A visual representation of the CNN-LSTM model architecture is provided in Figure 5.2, depicting the combination of convolutional and sequential learning components.

By leveraging CNNs for feature extraction and LSTMs for sequential dependency learning, this hybrid approach improves stock price forecasting by capturing both spatial and temporal patterns in the data.
The tabular presentation of model summary for both architectures have been provided in Table 1.

Table 2: Presentation of model summary.

Parameter	CNN Model	CNN-LSTM Model
Filters in First Convolutional Layer	64	64
Filters in Second Convolutional Layer	128	128
LSTM Units (Layer 1)	-	64
LSTM Units (Layer 2)	-	32
Kernel Size	Dynamically adjusted (1-3)	Dynamically adjusted (1-3)
Pooling Size	2 (or 1 for smaller steps)	2 (or 1 for smaller sequences)
Activation Function	ReLU	ReLU
Loss Function	Mean Squared Error (MSE)	Mean Squared Error (MSE)
Optimizer	Adam (learning rate = 0.0001)	Adam (learning rate = 0.0001)
Early Stopping (Patience)	3	3

1.29. Evaluation Metrics

1.29.1. Root Mean Squared Error

In deep learning, models assess their performance using a loss function, which measures how much the predicted values deviate from actual outcomes. Essentially, the loss function serves as a penalty mechanism for inaccurate predictions, where a loss value of zero signifies a perfect prediction. The fundamental goal during model training is to adjust weights and biases in a way that minimizes this loss as much as possible.

While loss functions primarily guide the training process, researchers often use Root Mean Squared Error (RMSE) as a standard evaluation metric to measure prediction accuracy. RMSE quantifies the deviation between predicted and actual values, providing a clear indication of how well the model performs. The formula for computing RMSE is as follows:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

In the equation, N represents the total number of data points, y_i denotes the actual observed value, and $\hat{y}_i$ refers to the predicted value. A key advantage of utilizing RMSE is its ability to heavily penalize significant prediction errors. Moreover, RMSE results are expressed in the same units as the target variable, making interpretation straightforward.

In this equation, N represents the total number of data points, y_i denotes the actual observed values, and $\hat{y}_i$ corresponds to the predicted values. One of the notable advantages of RMSE is its sensitivity to large errors, as it assigns greater penalties to substantial deviations. Additionally, since RMSE is expressed in the same units as the target variable, it facilitates intuitive interpretation, making it easier to assess the model's predictive accuracy.

1.30. Mean Absolute Percentage Error (MAPE)

Mean Absolute Percentage Error (MAPE) is a widely adopted metric for evaluating the accuracy of forecasting models. It measures the average percentage difference between predicted and actual values, offering a standardized way to assess model performance. The mathematical formula for MAPE is expressed as:

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right| * 100$$

Where:

N represents the total number of data points,

y_i denotes the actual observed values, and

$\hat{y}_i$ refers to the predicted values.

Since MAPE expresses errors as a percentage, it allows for easy interpretation and comparison across different datasets. A lower MAPE value indicates that the model produces more accurate predictions, whereas a higher MAPE suggests greater deviations between the predicted and actual values.

1.31. Coefficient of Determination (R^2)

The R-squared (R^2) metric is a widely used statistical measure that evaluates how well a predictive model explains the variance in the dependent variable. It assesses the model's goodness of fit by determining the proportion of variability in the actual values that can be explained by the predicted values. The mathematical expression for R^2 is given as:

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}$$

Where:

N represents the total number of observations,

y_i denotes the actual observed values,

$\hat{y}_i$ refers to the predicted values, and

$\bar{y}$ is the mean of all actual observed values.

The R^2 value ranges from 0 to 1, where a value closer to 1 signifies that the model effectively captures the variance in the dataset, leading to higher predictive accuracy. Conversely, an R^2 value near 0 suggests that the model provides little explanatory power, meaning it struggles to fit the observed data effectively.

2.. RESULTS AND ANALYSIS

This section presents a detailed performance comparison between the CNN model and the CNN-LSTM model based on three key evaluation metrics. The results based on evaluation metrics have been given in table 6.1

Table 3. Results based on evaluation metrics.

Models	CNN			CNN-LSTM		
Company	**R2**	**MAPE**	**RMSE**	**R2**	**MAPE**	**RMSE**
BAJAJ-AUTO	0.92	0.048	111.53	0.97	0.037	91.52
HDFCBANK	0.83	0.021	23.59	0.86	0.01	19.45
HINDALCO	0.90	0.019	8.16	0.97	0.013	6.52
HINDUNILVR	0.75	0.018	34.62	0.86	0.011	30.56
ICICIBANK	0.88	0.017	12.42	0.94	0.01	9.21
ITC	0.91	0.023	6.36	0.98	0.013	4.95
LT	0.92	0.021	52.52	0.99	0.013	41.50
RELIANCE	0.89	0.019	18.40	0.95	0.011	16.37
SBIN	0.82	0.018	11.01	0.88	0.013	10.40
TATAMOTORS	0.89	0.023	13.17	0.99	0.018	11.95
TATASTEEL	0.87	0.019	2.17	0.94	0.014	1.85

2.1. Model Performance Comparison

The results reveal a consistent improvement in predictive accuracy when using the CNN-LSTM model compared to the CNN model. The CNN-LSTM model effectively captures both spatial and temporal dependencies, making it a more robust approach for stock price forecasting.

2.2. R^2 Score Comparison

The CNN-LSTM model demonstrates higher R^2 values across all companies, indicating a better fit to the actual stock prices. Some notable improvements include:

HINDUNILVR: Increased from 0.75 (CNN) to 0.86 (CNN-LSTM), indicating a stronger correlation between predicted and actual prices.

ICICIBANK: Improved from 0.88 (CNN) to 0.94 (CNN-LSTM), showcasing enhanced predictive power.

TATAMOTORS: Increased from 0.89 (CNN) to 0.99 (CNN-LSTM), almost perfect predictability.

The higher R^2 values across all stocks confirm that the CNN-LSTM model provides a better representation of stock price movements, making it more reliable for forecasting.

2.3. MAPE Comparison

The Mean Absolute Percentage Error (MAPE) values are consistently lower in the CNN-LSTM model, showing that it produces predictions with lower percentage errors. Key improvements include:

1. HDFCBANK: MAPE decreases from 0.021 (CNN) to 0.010 (CNN-LSTM), reducing prediction error by more than 50%.
2. HINDUNILVR: Drops from 0.018 (CNN) to 0.011 (CNN-LSTM), showing better handling of stock price variations.
3. ICICIBANK: Reduces from 0.017 (CNN) to 0.010 (CNN-LSTM), making predictions more precise.

A lower MAPE value means that the CNN-LSTM model's predictions are closer to the actual prices, leading to more reliable stock forecasting.

2.4. RMSE Comparison

The CNN-LSTM model significantly lowers the RMSE values, indicating reduced absolute prediction errors. Some remarkable improvements include:

1. BAJAJ-AUTO: RMSE drops from 111.53 (CNN) to 91.52 (CNN-LSTM), meaning the CNN-LSTM model makes less extreme errors.
2. HINDALCO: Reduces from 8.16 (CNN) to 6.52 (CNN-LSTM), demonstrating finer precision.
3. LT: RMSE drops from 52.52 (CNN) to 41.50 (CNN-LSTM), showing the model's ability to make closer predictions to actual values.

Lower RMSE values confirm that the CNN-LSTM model produces more stable and less volatile predictions.

2.5. Loss Statistics Analysis

This section presents an in-depth analysis of the loss statistics for both the CNN model and the CNN-LSTM model, focusing on their performance during training and validation. Loss statistics are crucial indicators of model stability and generalization ability, helping assess how well a model minimizes errors over time. The Loss statistics for CNN model have been provided in table 6.2 and the loss statistics for CNN-LSTM have been provided in table 6.3.

Table 4: Loss statistics based on CNN model.

	Training statistics			Validation statistics		
Company	**Min**	**Max**	**SD**	**Val Min**	**Val Max**	**Val SD**
BAJAJ-AUTO	0.00025	0.285	0.113	0.004	0.009	0.002
HDFCBANK	0.00009	0.02	0.009	0.0015	0.0018	0.0001
HINDALCO	0.00023	0.011	0.003	0.0006	0.0029	0.0007
HINDUNILVR	0.00004	0.029	0.006	0.0008	0.0013	0.0001
ICICIBANK	0.00007	0.057	0.012	0.0007	0.0016	0.0002
ITC	0.00006	0.086	0.026	0.0007	0.002	0.0004
LT	0.00009	0.044	0.018	0.0014	0.0018	0.0002
RELIANCE	0.00006	0.048	0.012	0.0006	0.0014	0.0002
SBIN	0.00006	0.146	0.029	0.0005	0.0021	0.0003
TATAMOTORS	0.00016	0.091	0.022	0.0006	0.0018	0.0003
TATASTEEL	0.0001	0.027	0.006	0.0004	0.0012	0.0002

Table 5: Loss statistics based on CNN model.

	Training statistics				Validation statistics		
Company	**Min**	**Max**	**SD**	**Batches**	**Val Min**	**Val Max**	**Val SD**
BAJAJ-AUTO	0.00025	0.285	0.113	5	0.004	0.009	0.002
HDFCBANK	0.00009	0.02	0.009	4	0.0015	0.0018	0.0001
HINDALCO	0.00023	0.011	0.003	10	0.0006	0.0029	0.0007
HINDUNILVR	0.00004	0.029	0.006	20	0.0008	0.0013	0.0001
ICICIBANK	0.00007	0.057	0.012	22	0.0007	0.0016	0.0002
ITC	0.00006	0.086	0.026	10	0.0007	0.002	0.0004
LT	0.00009	0.044	0.018	5	0.0014	0.0018	0.0002
RELIANCE	0.00006	0.048	0.012	14	0.0006	0.0014	0.0002
SBIN	0.00006	0.146	0.029	24	0.0005	0.0021	0.0003
TATAMOTORS	0.00016	0.091	0.022	16	0.0006	0.0018	0.0003
TATASTEEL	0.0001	0.027	0.006	16	0.0004	0.0012	0.0002

Table 6: Loss statistics based on CNN-LSTM model.

	Training statistics			Validation statistics		
Company	**Min**	**Max**	**SD**	**Val Min**	**Val Max**	**Val SD**
BAJAJ-AUTO	0.00082	0.548	0.215	0.00749	0.155	0.059
HDFCBANK	0.00003	0.283	0.051	0.00092	0.134	0.024
HINDALCO	0.00022	0.417	0.073	0.00065	0.302	0.051
HINDUNILVR	0.00004	0.137	0.043	0.00141	0.012	0.003
ICICIBANK	0.00009	0.36	0.099	0.0007	0.008	0.002
ITC	0.00005	0.339	0.084	0.00084	0.031	0.008
LT	0.00004	0.399	0.08	0.0013	0.042	0.008
RELIANCE	0.00006	0.266	0.084	0.00065	0.007	0.002
SBIN	0.00006	0.326	0.084	0.00047	0.013	0.003
TATAMOTORS	0.00019	0.285	0.076	0.00062	0.008	0.002
TATASTEEL	0.00009	0.342	0.071	0.00047	0.005	0.001

2.6. Training Loss Analysis

The training loss statistics, including minimum, maximum, standard deviation (SD), and batch counts, provide insights into the learning process of both models.

2.7. CNN Model Training Loss

1. The minimum training loss values for all stocks remain very low, indicating that the model effectively reduces errors during training.
2. The maximum training loss varies significantly among different stocks, with BAJAJ-AUTO (0.285) and SBIN (0.146) showing the highest values. This suggests that certain stocks exhibit more volatility in the learning process.
3. The standard deviation (SD) values, such as 0.113 for BAJAJ-AUTO and 0.029 for SBIN, indicate varying levels of fluctuation in training loss.
4. The batch size distribution shows that the number of batches used for training differs among stocks, with SBIN (24 batches) and ICICIBANK (22 batches) requiring more iterations.

2.8. CNN-LSTM Model Training Loss

1. The CNN-LSTM model exhibits higher maximum loss values during training, with BAJAJ-AUTO (0.548) and HINDALCO (0.417) showing the highest loss peaks.
2. The standard deviation values are generally higher in the CNN-LSTM model, reflecting increased variability in training.
3. Despite higher training loss fluctuations, the lower minimum training loss values suggest that the model successfully learns the patterns of stock price movements over iterations.

4. Stocks like HDFCBANK (30 batches) and HINDALCO (33 batches) required more batches for convergence, implying that CNN-LSTM requires more data for effective learning.

2.9. CNN Model Validation Loss

1. The minimum validation loss values across all stocks are very small, with TATASTEEL (0.0004) and SBIN (0.0005) having the lowest loss values.
2. The maximum validation loss values, such as BAJAJ-AUTO (0.0090) and ITC (0.0020), suggest that while CNN maintains stable performance, certain stocks still experience noticeable prediction deviations.
3. The standard deviation values remain low across all stocks, indicating consistent performance across validation datasets.

2.10. CNN-LSTM Model Validation Loss

1. The CNN-LSTM model shows mixed results in validation loss. Some stocks exhibit higher maximum validation loss values, such as HINDALCO (0.302) and HDFCBANK (0.134), indicating occasional prediction instability.
2. However, stocks like TATASTEEL (0.005) and SBIN (0.013) show a significant reduction in maximum validation loss compared to CNN, confirming better generalization.
3. The standard deviation in validation loss is generally higher in CNN-LSTM compared to CNN, indicating greater variability in model performance across different stocks.

2.11. Insights Based on Comparative Results

The comparative analysis between CNN and CNN-LSTM models highlights the significant advantages of using a hybrid approach that combines convolutional and long short term memory neural networks. Based on the achieved results on evaluation metrics the following key insights can be inferred.

1. The CNN-LSTM model outperforms the CNN model in all key performance metrics, confirming its effectiveness in stock price forecasting.
2. The higher R^2 values show that the CNN-LSTM model better explains stock price variations, while the lower MAPE and RMSE values indicate higher accuracy and lower errors.
3. The CNN model focuses only on spatial dependencies but lacks the ability to track temporal trends effectively.
4. By integrating LSTM layers, the CNN-LSTM model remembers past price movements and uses that knowledge to predict future prices more effectively.
5. The lower MAPE and RMSE values in the CNN-LSTM model make it more suitable for practical stock market applications.
6. The CNN-LSTM model exhibits higher training loss variance, particularly for stocks like BAJAJ-AUTO and HINDALCO, suggesting a more complex learning process.
7. The larger batch size requirements in CNN-LSTM also indicate that this model demands more extensive training data to reach optimal performance.
8. The CNN model exhibits lower validation loss fluctuations across most stocks, indicating stable performance on unseen data.
9. However, CNN-LSTM achieves significantly lower minimum validation loss values in certain cases (e.g., SBIN and TATASTEEL), suggesting strong predictive accuracy for certain stocks.
10. The higher standard deviation in CNN-LSTM training loss suggests that it learns more dynamically but also struggles with stability.
11. The CNN model exhibits lower standard deviation values in both training and validation, confirming a more consistent learning trajectory but potentially at the cost of missing complex dependencies.
12. The CNN model maintains a more stable loss trend, making it more suitable for datasets with limited historical dependencies.
13. The CNN-LSTM model, while requiring more training time and larger batches, exhibits better generalization for certain stocks.
14. For stocks with strong sequential dependencies, such as SBIN and RELIANCE, the CNN-LSTM model provides better long-term forecasting accuracy.

Figures representing the comparison of actual and predicted values of stocks based on CNN-LSTM model have been provided here.

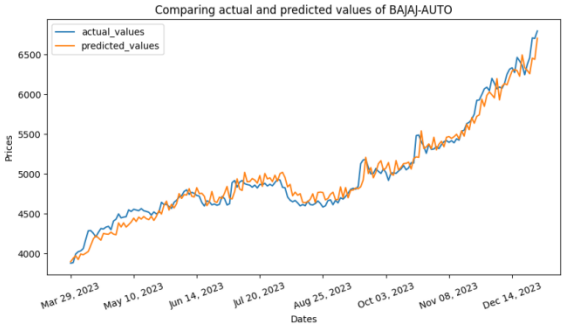

Figure 14: Values of Bajaj-Auto.

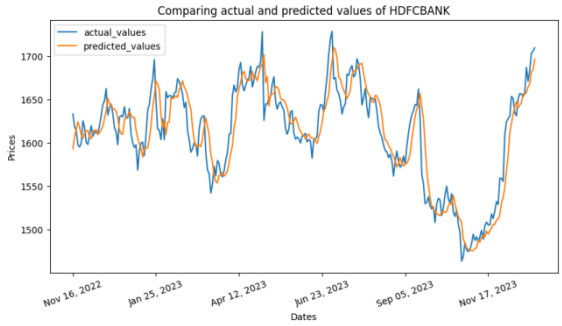

Figure 15: Values of HDFCBANK.

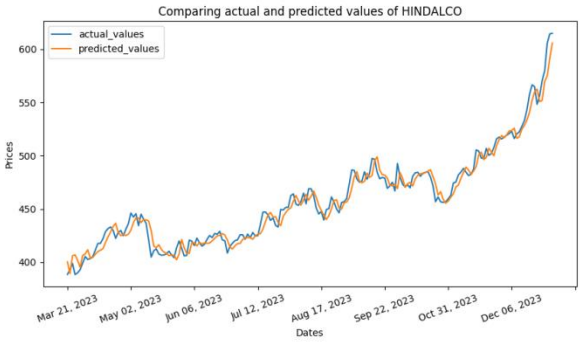

Figure 16: Values of HINDALCO.

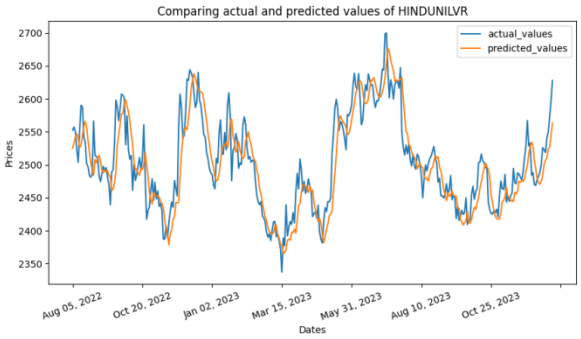

Figure 17: Values of HINDUNILVR.

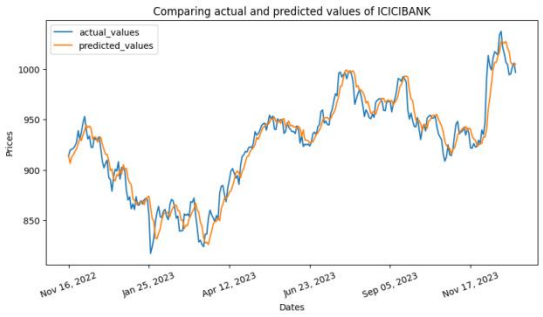

Figure 18: Values of ICICIBANK.

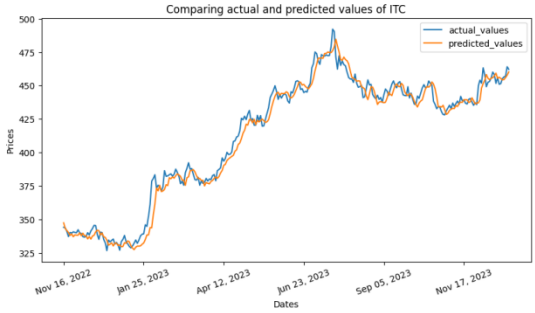

Figure 19: Values of ITC.

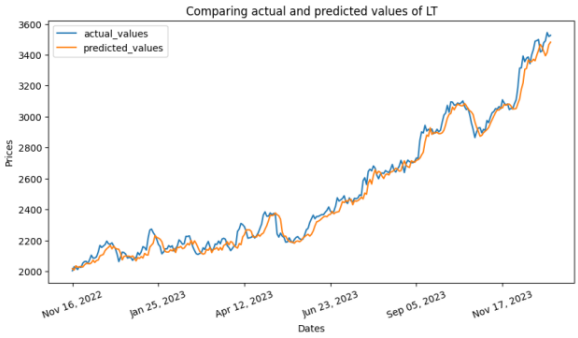

Figure 20: Values of LT.

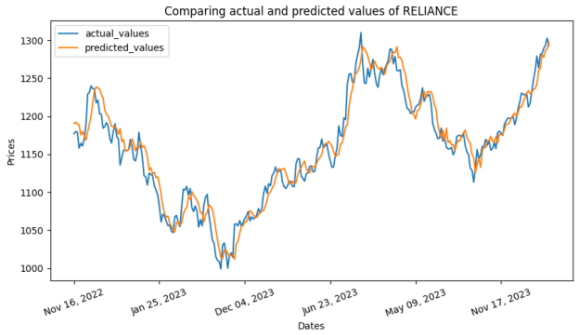

Figure 21: Values of RELIANCE.

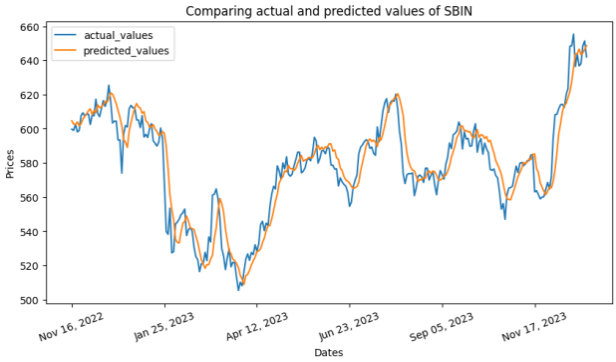

Figure 22. Values of SBIN.

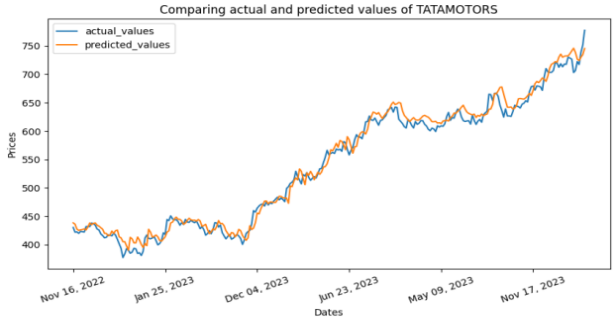

Figure 23: Values of TATAMOTORS.

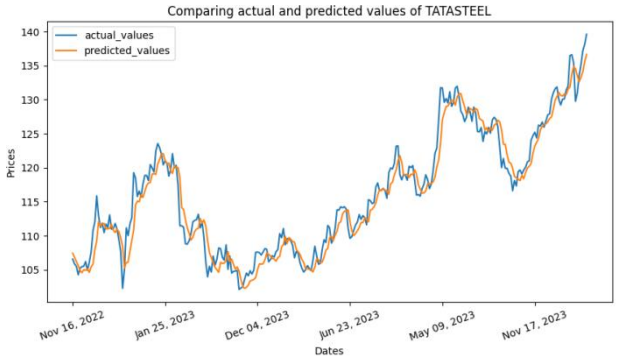

Figure 24: Values of TATASTEEL.

3. CONCLUSION

This study focused on predicting stock prices using historical OHLC (Open, High, Low, Close) data from 11 selected stocks which have been part of the NIFTY50 index since inception and were also present in the index as on 31-12-2023. The dataset was carefully preprocessed, including data cleaning, transformation, and scaling, to ensure high-quality input for the deep learning models. The time-series nature of stock prices posed a challenge, which was addressed by leveraging advanced deep learning architectures. We utilized 1D convolutional layers to extract spatial patterns in stock price movements. We also combined CNN for feature extraction with LSTM for sequential learning, enhancing predictive capabilities. The following key insights were found.

1. R^2 Score: The CNN-LSTM model outperformed the CNN model, achieving higher R^2 values (above 0.95 for several stocks), indicating a better fit.
2. MAPE: The CNN-LSTM model consistently lowered percentage errors, confirming higher accuracy in stock price predictions.
3. RMSE: The CNN-LSTM model reduced absolute prediction errors, leading to more precise forecasts.
4. Loss Statistics: The CNN model exhibited stable training trends, whereas the CNN-LSTM model, despite higher training fluctuations, generalized better in the validation phase.

These results confirm that while CNN models offer stability, the CNN-LSTM hybrid model provides superior predictive accuracy, making it better suited for financial forecasting.

3.1. Implications and Future Suggestions

1. The CNN-LSTM model's ability to capture both short-term and long-term dependencies makes it a valuable tool for portfolio optimization and risk assessment.
2. The model can be integrated into automated trading systems to improve real-time decision-making and reduce market risks.
3. Given its performance on stock data, the model can be extended to other financial assets, such as commodities, forex, and cryptocurrencies.
4. Future research can enhance prediction accuracy by incorporating macroeconomic indicators, investor sentiment analysis, and global market trends.
5. Implementing Attention-based CNN-LSTM architectures can improve model explainability and accuracy.
6. Combining deep learning with reinforcement learning can help develop adaptive trading strategies that evolve with market trends.
7. Testing the model on tick-level financial data could enhance its short-term price prediction capabilities.
8. Developing interpretable AI models can provide transparency, allowing financial analysts and investors to better understand model predictions.

This study highlights the potential of deep learning in financial forecasting, with the CNN-LSTM model proving to be an effective predictive tool. With further advancements, such models can play a crucial role in shaping AI-driven financial decision-making.

REFERENCES

A. Aadhitya, R. Rajapriya, V. R. S., and A. M. Bagde, "Predicting Stock Market Time-Series Data using CNN-LSTM Neural Network Model," arXiv preprint arXiv:2305.14378, 2023.

A. Adebiyi, A. O. Adewumi, and C. K. Ayo, "Stock Price Prediction Using the ARIMA Model," 16th International Conference on Computer Modeling and Simulation (UKSim-AMSS), 2014.

A. M. Alonso and C. Garcia-Martos, "Time Series Analysis – Forecasting with ARIMA Models," Universidad Carlos III de Madrid, Universidad Politecnica de Madrid, 2012.

A. P. Wibawa, A. B. P. Utama, H. Elmunsyah, U. Pujianto, F. A. Dwiyanto, and L. Hernandez, "Time-series analysis with smoothed Convolutional Neural Network," Journal of Big Data, vol. 9, no. 1, p. 44, 2022.

A. Staffini, "Stock price forecasting by a deep convolutional generative adversarial network," Frontiers in Artificial Intelligence, vol. 5, p. 837596, 2022.

C. R. Cao, "Stock Price Prediction Using Deep-Learning Models: CNN, RNN, and LSTM," SHS Web of Conferences, vol. 196, p. 02004, 2024.
G. Box and G. Jenkins, Time Series Analysis: Forecasting and Control, San Francisco: Holden-Day, 1970.
G. Zhang, T. Ren, and Y. Yang, "A New Unified Deep Learning Approach with Decomposition-Reconstruction-Ensemble Framework for Time Series Forecasting," arXiv preprint arXiv:2002.09695, 2020.
J. Brownlee, "How to Create an ARIMA Model for Time Series Forecasting with Python," Machine Learning Mastery, 2017.
J. Brownlee, Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras, Machine Learning Mastery, 2016.
J. Kim and N. Moon, "BiLSTM model based on multivariate time series data in multiple fields for forecasting trading area," Journal of Ambient Intelligence and Humanized Computing, pp. 1-10, 2020.
J. Patterson, Deep Learning: A Practitioner's Approach, O'Reilly Media, 2017.
J. Schmidhuber, "Deep learning in neural networks: An overview," Neural Networks, vol. 61, pp. 85-117, 2015.
K. Chen and Y. He, "Stock Prediction Using Convolutional Neural Network," IOP Conference Series: Materials Science and Engineering, 2018.
K. Pardeshi, S. S. Gill, and A. M. Abdelmoniem, "Stock Market Price Prediction: A Hybrid LSTM and Sequential Self-Attention Based Approach," arXiv preprint arXiv:2308.04419, 2023.
K. Tatane, M. R. Sahib, and T. Zaki, "From Time Series to Images: Revolutionizing Stock Market Predictions with Convolutional Deep Neural Networks," International Journal of Advanced Computer Science & Applications, vol. 15, no. 1, 2024.
M. J. Kane, N. Price, M. Scotch, and P. Rabinowitz, "Comparison of ARIMA and Random Forest Time Series Models for Prediction of Avian Influenza H5N1 Outbreaks," BMC Bioinformatics, vol. 15, no. 1, 2014.
M. Khashei and M. Bijari, "A novel hybridization of artificial neural networks and ARIMA models for time series forecasting," Applied Soft Computing, vol. 11, no. 2, pp. 2664-2675, 2011.
Q. Li, N. Kamaruddin, S. S. Yuhaniz, and H. A. A. Al-Jaifi, "Forecasting stock prices changes using long-short term memory neural network with symbolic genetic programming," Scientific Reports, vol. 14, no. 1, p. 422, 2024.
R. J. Hyndman, "Variations on Rolling Forecasts," 2014.
R. J. Hyndman and G. Athanasopoulos, Forecasting: Principles and Practice, OTexts, 2014.
S. Mehtab and J. Sen, "Robust Analysis of Stock Price Time Series Using CNN and LSTM-Based Deep Learning Models," arXiv preprint arXiv:2011.08011, 2020.
S. Sang and L. Li, "A Novel Variant of LSTM Stock Prediction Method Incorporating Attention Mechanism," Mathematics, vol. 12, no. 7, p. 945, 2024.
S. S. Namini, N. Tavakoli, and A. S. Namin, "A Comparison of ARIMA and LSTM in Forecasting Time Series," 17th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 1394-1401, IEEE, 2018.
T. Fischer and C. Krauss, "Deep Learning with Long Short-Term Memory Networks for Financial Market Predictions," FAU Discussion Papers in Economics 11, Friedrich-Alexander University, 2017.
Y. Wang and I. Choi, "Market Index and Stock Price Direction Prediction using Machine Learning Techniques: An Empirical Study on the KOSPI and HSI," Technical Report, 2013.
Z. Cui, R. Ke, and Y. Wang, "Deep Stacked Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-Wide Traffic Speed Prediction," arXiv preprint arXiv:1801.02143, 2018.
Z. Li, J. X. Jensen, and Y. Mo, "Stock Market Analysis and Prediction Using LSTM: A Case Study on Technology Stocks," Innovations in Applied Engineering and Technology, 2023.