I.J. Image, Graphics and Signal Processing, **2025, 3, 104-122**
Published Online on June 8, 2025 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijigsp.2025.03.06

Next-Gen Market Predictor: Transformed Moving Average Fast-RNN Hybrid with Advanced CNNS

Swarnalata Rath*
Department of Computer Science and Engineering, Siksha 'O' Anusandhan (Deemed to be University), Bhubaneswar, Odisha, 751030, India
Email: Swarnalata.rath@gmail.com
ORCID iD: https://orcid.org//0000-0003-0318-1065
*Corresponding Author

Nilima R. Das
Department of Computer Application, Siksha 'O' Anusandhan (Deemed to be University), Bhubaneswar, Odisha, 751030, India
Email: nilimadas@soa.ac.in
ORCID iD: https://orcid.org//0009-0008-1007-4376

Binod Kumar Pattanayak
Department of Computer Science and Engineering, Siksha 'O' Anusandhan (Deemed to be University), Bhubaneswar, Odisha, 751030, India
Email: binodpattanayak@soa.ac.in
ORCID iD: https://orcid.org//0009-0002-0670-3578

Received: 25 September, 2023; Revised: 16 October, 2023; Accepted: 12 January, 2025; Published: 08 June, 2025

Abstract: Stock price prediction anticipates future stock prices using historical data and computational models to assist and guide investing decisions. In financial forecasting, accuracy and efficacy in stock price prediction are essential for making better choices. This research describes a hybrid deep learning strategy for improving the extraction and interpretation of the crucial details from stock price time series data. Traditional approaches confront challenges such as computational complexity and nonlinear stock prices. The suggested method pre-processes stock data with Moving Average Z-Transformation, which emphasises long-term trends and reduces fluctuations in the short term. It combines a Transformed Moving Average Fast-RNN Hybrid with Advanced CNNs to create an efficient computational framework. The Enhanced Deep-CNN layer comprises convolutional layers, batch normalisation, leaky ReLU activations, dropout, max pooling and a dense layer. The performance of the model is quantified using metrics including Mean Absolute Error (MAE), Mean Square Error (MSE), Root Mean Square Error (RMSE), and R-squared (R^2). It shows superior prediction accuracy with MAEs of 0.28, 0.15, 0.34, 0.17, and 0.13 for Kotak, ICICI, Axis, and SBI, respectively, outperforming previous models. These measurements provide detailed information about the model's predictive skills, proving its ability to improve stock price forecast accuracy significantly.

Index Terms: Stock Price, Moving Average Z-Transformation, Fast-RNN, Deep CNN, Leaky Relu Activations, MSE

1. Introduction

The performance of a country's financial market is an essential factor in determining its overall economic status, allowing economists and financial specialists to assess the country's present economic health. The stock market is a crucial contributor among the many financial markets. A country's economic status impacts various industries, including finance, agriculture, metals, and investment banking. These sectors' development depends on their volatility, which adheres to the fundamental principle of supply and demand. Increased supply prompts traders and financial institutions to invest in that sector or stock, causing prices to rise [1]. Regular dividend payments also help to generate profits and returns on invested capital. Investors must find the best time to sell shares and obtain their desired profits. Financial

markets include stocks, derivatives, bonds, and commodities [2]. The stock market allows investors to invest in companies and possess a portion or percentage. As a company grows, it frequently requires additional cash to finance its future operations. Companies can sell these shares to investors after receiving approval from present shareholders, who will erode their shareholding due to the issuance of additional shares. Successful outcomes lead to enhanced stock market value for the shares.

Stock price indicators serve as fundamental benchmarks in financial markets, shaping investment strategies and market dynamics worldwide. These indicators aggregate the intricate movements of individual company stocks or specific market segments on exchanges, reflecting the collective sentiment and underlying economic conditions [3]. Forecasting these indicators, especially stock price movements, poses a formidable challenge for stakeholders and researchers, given their pivotal role in investment decision-making. Accurate forecasts of stock price changes have become essential for investors trying to maximise their portfolio performance [4]. The ability to anticipate whether a stock's price will rise or fall informs decisions to buy, sell, or hold securities, influencing profitability and risk management strategies. At its core, stock price prediction aims to identify stocks with a high likelihood of price appreciation for purchase and those with a high probability of price depreciation for sale, aligning investment decisions with anticipated market trends.

Two primary approaches to stock market forecasting have emerged: fundamental and technical analysis. Fundamental analysis delves into an organisation's intrinsic value by examining market position, financial performance, costs, revenue, and growth prospects. By assessing these essential data points, investors may determine an organisation's inventory's underlying strength and probable future trajectory, guiding their investing decisions accordingly. In contrast, technical analysis examines past price movements and market trends to forecast potential price behaviour [5]. This method is based on the notion that past price patterns repeat themselves and can be used to predict price movements in the future [6, 7, 8]. Technical analysts use charts, graphs, and indicators of trends, such as moving averages and trade volumes, to discover patterns and trends that indicate probable buy or sell signals, which aids in developing trading strategies.

Time series decomposition is a frequent technique used in stock price forecasting. This entails breaking down past stock price data into essential components, such as trend, seasonality, and random variations [9]. By isolating these components, analysts can identify underlying patterns and trends that inform future price movements, facilitating more accurate predictions and informed decision-making. Despite the advancements in forecasting techniques, stock price prediction remains challenging due to financial markets' inherent uncertainty and complexity [10]. Factors such as market sentiment, geopolitical events, and macroeconomic trends introduce volatility and unpredictability, complicating accurately forecasting price movements.

Machine learning has also significantly affected stock price prediction, as it allows for analysing large datasets and discovering patterns that are difficult to detect using traditional methods. Stock prices have been predicted using techniques such as SVM, decision trees, and ensemble learning approaches, which learn from previous data and recognise complicated patterns [11]. ML models may process various inputs, including stock prices, technical indications, and economic considerations, to create predictive models that help investors make better decisions [12]. However, these models frequently necessitate substantial feature engineering and need to be improved because of their inability to learn hierarchical representations of data automatically.

Deep learning has become a potent instrument in stock price prediction, providing benefits above conventional statistical techniques and fundamental analysis [13]. Unlike traditional models, which often require manual feature engineering and assumptions about underlying data distribution, DL models can learn complicated patterns and representations from the raw data. One of its main advantages is deep learning's capacity to manage high-dimensional data well. Deep learning models can process large amounts of information regarding stock price prediction, including historical price data, trade volumes, news mood, macroeconomic indicators, and more [14]. By ingesting such diverse sources of information, deep learning models can capture subtle relationships and interactions that may not be apparent to human analysts or traditional statistical models. Hybrid deep learning techniques combine the strengths of deep learning models with other methodologies, such as conventional statistical methods or expert knowledge, to improve prediction accuracy further. These hybrid techniques combine the interpretability of classical models with the tremendous feature extraction capabilities of DL systems.

Literature suggests a two-hybrid method using RNN-LSTM, FPA, and PSO to generate the most excellent DL model for real-time stock market prediction. The two-hybrid approach uses a methodical technique to create optimum RNN-LSTM networks [15]. Simulated Annealing (SA), Particle Swarm Optimization (PSO), Single-Guided-PSO (SGPSO) [16], Genetic Algorithms (GA), Ant Colony Optimization (ACO), and Artificial Bee Colony (ABC) are a few of the most effective metaheuristic algorithms. Some more recent approaches, like the Grey Wolf Optimization (GWO), Harris' Hawks Optimizer (HHO), Bacterial Foraging Optimization (BFO), Moth-Flame Optimization (MFO), Salp Swarm Algorithm (SSA), and Whale Optimization Algorithm (WOA), have also attracted sufficient interest from researchers due to their effectiveness in solving problems [17]. The recently developed Equilibrium Optimizer (EO) [18] draws inspiration from models of managing volume mass balance utilised to forecast equilibrium and dynamic situations. This study mainly focuses on creating an optimised hybrid approach that can forecast stock prices correctly for the subsequent day of trading in a short period by examining the tremendous effectiveness of DL-based hybrid approaches in diverse domains for stock price prediction. The primary accomplishments of this proposed work are given below:

- Stock price prediction is improved by using a novel hybrid approach that combines various deep neural network components.
- The Moving Average Z-Transformation approach is used for sophisticated data pre-processing to highlight long-term patterns while reducing short-term oscillations, resulting in better data analysis and interpretation.
- To address computational challenges and improve scalability, an efficient computational framework based on the Transformed Moving Average Fast-RNN Hybrid model is used.
- Convolutional layers, batch normalisation, Leaky ReLU, dropout, max pooling and fully connected layer have all been added to Deep-CNN layer development to improve feature retention and reduce complexity.

The remaining portion of the document is organised as follows: Section 2 reviews the literature on DL and ML-based models used for stock market prediction. This section focuses on both single-network and hybrid models. Section 3 thoroughly explains the proposed models and the mathematical model and structural alterations used to implement them. Section 4 describes the experiment's setup, datasets, and simulation results for predicting stock prices. In Section 5, the work's conclusion is presented.

2. Literature Survey

The use of DL algorithms has resulted in substantial advancements in financial forecasting. Several studies have investigated various neural network designs and pre-processing approaches for improving the accuracy and robustness of stock price forecasting models. The following section outlines the primary methodologies and procedures used in financial forecasting and its pros and cons.

Liang et al. [19] provided an example of how candlestick patterns and sequence similarity were utilised to predict trends in a stock time series. It used sequential pattern mining to extract patterns from multidimensional data and innovative sequence similarity approaches to match previously existing patterns. Although the technique beat SVM and LSTM models, it required more refinement to account for all stocks.

Wang et al. [20] developed a hybrid technique for stock forecasting trends based on ELM and wavelet transform denoising. DWT was used to denoise raw data, which was then pre-processed for features and DELM training. The DELM was then compared against the original ELM model on a dataset of 400 equities. The findings revealed that the DELM significantly increased accuracy and AUC metrics, while the wavelet transform might improve the ELM model's prediction abilities. The logical relationship between DWT-based denoising and the labelling method was also investigated. The DELM surpassed 12 typical algorithms in predicting outcomes, demonstrating its superiority. However, the article did not examine the impact of various wavelet function denoising findings on stock trend prediction accuracy.

Tao et al. [21] developed an innovative technique for stock price forecasting by using a Conv LSTM network, DL algorithm, and knowledge graph. They combined market data to predict future stock values, mined hidden links between stocks and created a mutation point distance weight matrix. However, missing values and dataset noise impacted the model's performance.

Dingming et al. [22] the authors suggested a hybrid framework that combined discrete wavelet transform (DWT), extreme learning machine (ELM), and auto-encoder (AE) with feature penalty. After denoising the raw data with DWT, the theoretical backpropagation of the AE with feature penalty was calculated. A denoising-AE-ELM (DAELM) hybrid framework was successfully constructed, with the well-trained AE's encoder component used to train the ELM model via feature extraction. The results demonstrated the higher performance of the proposed hybrid framework. However, utilising a mix of equities to develop a successful portfolio was emphasised.

Vo et al. [23] compared the effects of LSTM, Bi-LSTM, and gated recurrent units on stock price prediction. They discovered that the Bi-LSTM model read the data one more time backwards, which improved prediction accuracy, particularly for sequential data such as financial time series.

Kanwal et al. [24] suggested a hybrid DL model that used a 1D CNN and a Bidirectional Cuda DNN LSTM to predict stock prices. They used five stock price datasets to compare the proposed model to other hybrid DL-based and cutting-edge models. Findings indicated that the proposed hybrid technique performed well in terms of dependable and precise stock price forecasting, assisting investors in making well-informed investment decisions. However, the research needed to focus on extracting more useful characteristics from multivariate datasets and developing a hybrid forecasting method based on multiple DL model types suitable for various time series forecasting applications such as weather, earthquakes, and gold prices.

There are two primary areas in which the literature requires sufficient study. To develop hybrid DL models for stock price prediction, new feature extraction approaches are needed, as well as the addition of other significant variables such as economic data and market sentiment. Second, although these DL models demonstrate promise in stock price forecasting, there is still much to be discovered about their potential in other time series forecasting domains, such as weather, seismic activity, and commodities pricing. The resilience and dependability of the models in real-world applications can also be increased by more research into data preparation methods and the effects of dataset noise. The following section provides a detailed explanation of the proposed technique.

3. Proposed Methodology

A novel hybrid deep learning approach is proposed to address significant challenges in stock price prediction, such as dealing with non-linear and volatile data, relying on historical patterns, which frequently limit accuracy, gradient stability issues during training, and the need for enhanced feature extraction beyond closing prices. This technology combines many specialised deep neural network components to extract crucial information from stock price time series data. It starts with pre-processing utilising the Moving Average Z-Transformation approach, which emphasises longer-term patterns while diminishing short-term swings, improving data interpretability and significantly lowering noise in raw stock data. The method employs a Transformed Moving Average fast RNN hybrid with Advanced CNNs to reduce computational cost and increase model generalisation on massive datasets.

This integration takes advantage of the Fast-RNN model's efficient computational architecture, allowing faster training and inference while improving scalability and performance. The Enhanced Deep-CNN layer is critical, using convolutional layers for local feature extraction, batch normalisation to stabilise training, Leaky ReLU activations for optimised gradient flow, dropout to prevent overfitting, and max pooling to retain important features while reducing computational complexity. The retrieved features are subsequently transformed into a higher-dimensional representation using fully linked layers, culminating in a dense layer with a single neuron for accurate stock price prediction. The model's effectiveness is assessed using evaluation measures such as MAE, MSE, RMSE, and R^2, which provide detailed insights into its predictive powers and accuracy in financial forecasting contexts. The workflow diagram of the proposed work is shown in Fig. 1.

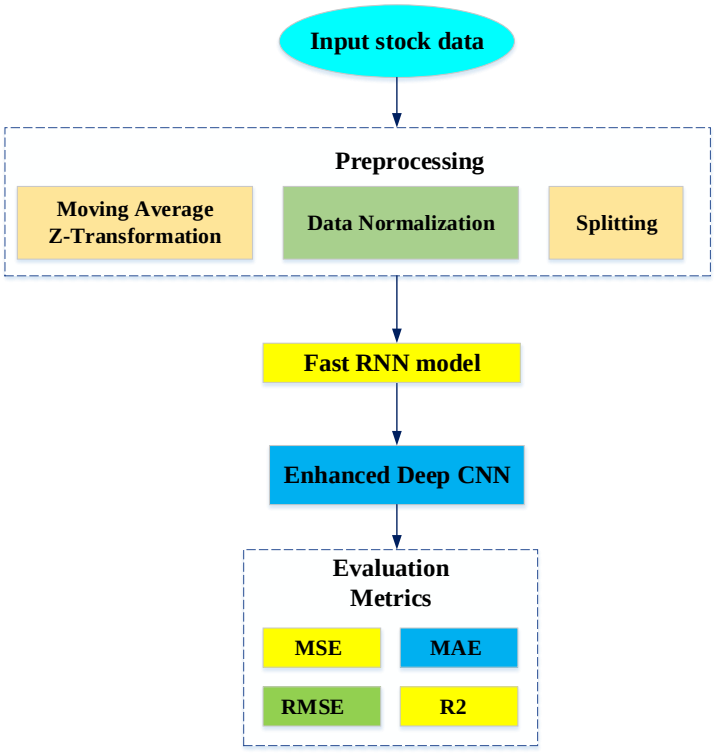


Fig. 1. Workflow diagram for proposed work

The final output forecast for stock prices is obtained by mapping the characteristics to a higher-dimensional representation using fully connected layers. Measures like MAE, MSE, RMSE, and R^2 evaluate the model's efficiency and accuracy and offer insights into its predictive power.

3.1 Pre-Processing

To begin, the input stock data undergoes pre-processing using the Moving Average Z-Transformation method to emphasise longer-term patterns in data and reduce short-term fluctuations. Additionally, it enables characteristics to be rescaled. A positive normalised number signifies that the x value exceeds the mean, whereas a negative score denotes a lower x value. This technique integrates two approaches: the simple moving average technique and data normalisation by Z-score normalisation. This method effectively smooths out noise and short-term fluctuations in raw stock price data, facilitating better analysis and interpretation of stock price movements.

The Simple Moving Average smoothes out short-term swings while emphasising longer-term patterns or cycles in stock information. It reduces noise and makes underlying trends more visible. This is beneficial in time series data,

where random variations obscure meaningful patterns. By smoothing the data, SMA can help reduce the impact of outliers and extreme values, leading to more stable and reliable data.

Determine the window size (W): Choose a period over which the average will be calculated.

Calculate the SMA: For each data point t, compute the average of data points in the window. The formula for SMA at time t is given in equation (1):

$$SMA_t = \frac{1}{W}\sum_{i=0}^{W-1} X_{t-i} \tag{1}$$

Where X is the original data series, replace the original data with the SMA values to get the smoothed data series.

Z-score transformation is a technique for normalising SMA-transformed data values, resulting in a mean of 0 and a standard deviation of 1. This is accomplished using the following formula. The mean (μ) and standard deviation (σ) of the SMA data series are determined in Equation (2) (3).

$$\mu = \frac{1}{N}\sum_{i=1}^{N} SMA_i \tag{2}$$

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(SMA_i - \mu)^2} \tag{3}$$

Apply the Z-score formula to each SMA value in equation (4):

$$Z_i = \frac{SMA_i - \mu}{\sigma} \tag{4}$$

Compute the SMA for the entire stock price series using a chosen window size to smooth the data and emphasise longer-term trends. Standardise the SMA-transformed data by determining the mean and standard deviation of the SMA series, then convert each data point to its associated Z-score. This technique significantly smoothing out short-term fluctuations, reduces noise, and reduces the impact of outliers and extreme values, leading to more stable data. This technique also standardises the data to a standard scale, enhancing algorithms' performance and convergence speed. The MAZT method effectively smooths out noise and short-term fluctuations while rescaling the data for improved analysis and interpretation. It is a powerful pre-processing technique for stock price data.

3.1.1 Data Splitting

The initial dataset is pre-processed before being separated into two distinct sets

(1) The training set of information used to develop the method
(2) The validation testing dataset that is used.

We split the dataset in two in our experiments, using 20% for testing and 80% for training.

3.2 Transformed moving average Fast-RNN hybrid with advanced CNNs

The proposed method combines the enhanced feature extraction capabilities of CNNs with the time series forecasting performance of Fast-RNN after pre-processing. Convolutional layers, batch normalisation, Leaky ReLU activation, dropout and max pooling are all included in the Enhanced Deep-CNN layer. Fully connected layers then map the retrieved features to a higher-dimensional representation, which results in the stock price forecast as the final output.

3.2.1 Fast RNN-based Hybrid Model

Stock price prediction techniques produce less accurate and dependable predictions due to their limitations, which include capturing underlying patterns with fully connected architectures, vanishing or exploding gradients during training, and delays in model generalisation caused by focusing on the latest characteristics during weight updating. These drawbacks are addressed by the Fast-RNN Hybrid model, which incorporates a practical computational framework that enables quick training and inference on large data sets. By utilising Fast-RNN, problems with vanishing and exploding gradients are lessened, guaranteeing more reliable and efficient training. Furthermore, combining the sophisticated CNN skills for extracting complex features with the strengths of RNNs for processing sequential data, the hybrid technique improves predictive accuracy and pattern recognition. Thanks to this integration, the model can generalise well even with big and complicated datasets, enhancing scalability and computing efficiency.

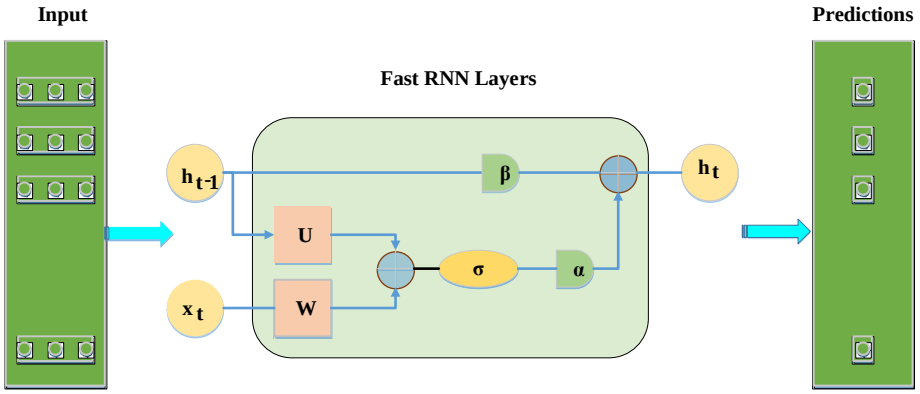


Fig. 2. Fast RNN Model

The model was modified by including residual layers, and it depends on the visual geometric group model, as depicted in Fig. 2. There are 25.5 million parameters used to train ResNet. As a result, ResNet has become a very complex network. Using ResNet's learnt residual connections, Fast RNN offers stable training at a significantly lower computational cost while maintaining comparable training accuracies. The accuracy of predictions made by Fast RNN is 19% greater than that of typical RNN structures and marginally lower than that of gated RNN structures. Typically, hidden state vectors in an RNN architecture are represented in equation (5):

$$h_t = tanh\left(W_{x_t} + Uh_{t-1} + b\right) \tag{5}$$

Where $x_t \in R^D$ denotes the t^{th} step feature vector. Learning U and W in the structure above is difficult because the gradient can have an exponentially large condition number (in T). To address this issue, unitary network-based approaches can enlarge the network, but doing so would make it noticeably larger and may affect the model's accuracy.

On the other hand, Fast RNN uses a standard weighted residual connection to produce well-conditioned gradients that stabilise the training. Specifically, Fast RNN makes the following modifications to the hidden state h_t using equation (6) (7):

$$h_t = \propto . h'_t + \beta . h_{t-1} \tag{6}$$

$$h'_t = \sigma\left(W_{x_t} + Uh_{t-1} + b\right) \tag{7}$$

The sigmoid function is used to parameterise the trainable weights α and β, which can range from 0 to 1. σ, like the sigmoid, hyperbolic tangent, or ReLU, is a nonlinear function that varies with the dataset.

A determined α, β limits the amount that the current attribute vector x_t can update the hidden state, which is how fast RNN reconditions the hidden state. Fast RNN takes less computations and has two more parameters than RNN, which is minor compared to the RNN's per-stride computational complexity. Fast RNN scales effectively to large datasets with conventional optimisation techniques because, in contrast to unitary methods, it does not impose any expensive systemic limits on U. Fig.2 displays the suggested Fast RNN-based model for stock market forecasts.

The optimised weight update mechanism, gate mechanisms, and backpropagation through Fast RNN time enhance training and inference efficiency. They are appropriate for big datasets and real-time applications, efficiently manage sequential data, and optimise algorithms. Rapid RNNs can be integrated with other neural network elements to combine spatial feature extraction with temporal sequence processing, which enhances prediction performance.

3.2.2 Enhanced Deep-CNN model

The Enhanced Deep-CNN layer, which consists of multiple layers intended to handle the non-linearity and high volatility of stock price data, comes after the Fast RNN. The disadvantages of conventional approaches include overfitting because of fully connected structures, problems with vanishing and bursting gradients, and difficulty capturing complicated patterns. Predictions of stock prices become less reliable and precise due to these difficulties. The suggested approach, which combines Fast RNN with sophisticated CNN functionalities, gets over these drawbacks.

Convolutional layers are used in the Enhanced Deep-CNN layer to extract patterns effectively. BN is used for stable training, Leaky ReLU activation is used to increase gradient flow, dropout is used to lessen overfitting, and max pooling is used to retain features. Together, these layers extract and process complex features that improve the model's capacity to identify subtle trends in stock price data. Accurate stock price predictions are produced by fully connected layers mapping the retrieved characteristics to a higher-dimensional representation. This hybrid strategy raises predictive accuracy by reducing computational inefficiencies, enhancing scalability, and guaranteeing more reliable and efficient training. The flow chart of Enhanced deep CNN is illustrated in Fig. 3.

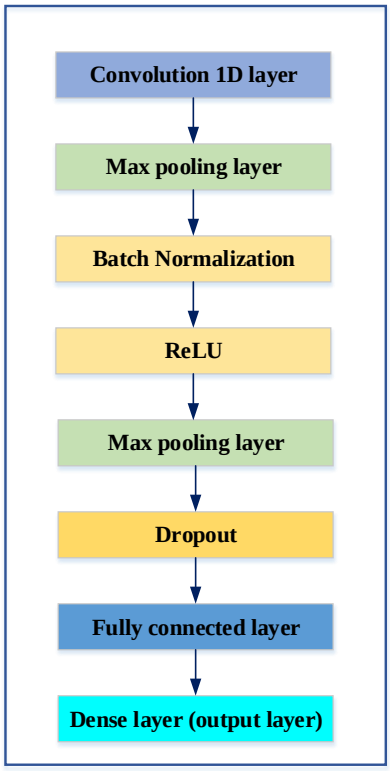


Fig. 3. Flowchart of Enhanced Deep CNN

3.2.2.1 Convolution Layer

Convolutional layers recognise and learn spatial hierarchies to identify local patterns and aspects of data, such as stock price fluctuations. They are perfect for jobs involving time series and images because they maintain the spatial structure of the incoming data. Convolutional layers use shared weights and local connections to get around the drawbacks of fully linked systems. They concentrate on minute, localised patterns and characteristics combined at lower levels to create more intricate representations. The mathematical expression for the convolution operation is determined in equation (8):

$$(f * g)(t) = \sum_{\propto=-\infty}^{\infty} f(a)g(t-a) \tag{8}$$

In this case, f stands for the input signal, g for the filter or kernel, and t for the calculation point of the convolution. All values of a, which stand for the input signal components, are added together. By swiping the filter g across the input signal f, element-wise multiplications are carried out, and the output is generated by adding the results.

3.2.2.2 Batch Normalization

BN is the next step after the convolutional layers extract local patterns and features to create feature maps. It is a deep learning strategy stabilising the training process by levelling filter activations across mini-batch sizes. It tackles difficulties such as learning rate problems, internal covariate shift, gradient vanishing/exploding, and sensitivity to initiation. By keeping the input mean and variance constant, BN reduces the need for network adaptation, preserves gradients, and lowers the network's sensitivity to weight initialisation. It also permits increased learning rates without running the risk of divergence. The mini-batch mean and variance are determined, and the inputs have been normalised utilising these statistics. The normalised values are then scaled and moved using learnable parameters in the batch normalisation process. Precisely, for a mini-batch $B = \{x_1, x_2, \ldots . x_m\}$, the mean$(\mu_B)$ and variance(σ_B^2) are calculated in equation (9) (10),

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{9}$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2 \tag{10}$$

Each input x_i in the mini-batch is normalised in equation (11) using the mean and variance:

$$\widehat{x_i} = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{11}$$

Here, ϵ is a minor constant added to ensure numerical stability. Normalised variables are scaled as well as shifted using parameters that are learned (γ and β), resulting in equation (12):

$$y_i = \gamma \widehat{x_i} + \beta \tag{12}$$

Where $\gamma \; and \; \beta$ are learned during training, enabling the model to adjust the normalised values. BN may increase processing costs and require precise tuning, but it stabilises training in RNN. Using the Leaky ReLU activation function enhances model performance.

3.2.2.3 Leaky ReLU activation

After batch normalisation, the network usually applies a non-linear activation function, like Leaky ReLU. When the unit is not in use, Leaky ReLU allows for a little, non-zero gradient, which makes it better than conventional ReLU. This guarantees improved gradient flow during training and helps to mitigate the problem of dead neurons, which occur when neurons stop learning because their gradients become zero. The function of Leaky ReLU is given in equation (13):

$$f(x) = \begin{cases} x & if \; x > 0 \\ \propto x & if \; x \leq 0 \end{cases} \tag{13}$$

Where $\propto$ is a small constant that allows a small, non-zero gradient when x is negative?

Leaky ReLU solves the dying ReLU problem by allowing a modest gradient when the input is negative. This guarantees that gradients flow through the network during backpropagation, resulting in more reliable and effective training. This is especially significant for deep networks since learning effectively depends on preserving gradient flow. While the Leaky ReLU activation function reduces the number of dead neurons and improves gradient flow, it does have some limitations. One of the disadvantages is that it does not naturally lower model complexity, which can lead to overfitting, particularly in intense networks. The parameter α must be carefully chosen to prevent the dying ReLU problem. The following process involves dropping out to address these restrictions.

3.2.2.4 Dropout

Dropout is a technique for randomly dropping a fraction of neurons during training, which helps to reduce overfitting and improve model generalisation. The dropout function can be described in equation (14):

$$y = \frac{1}{1-p} \sum_{i=1}^{n} x_i \, . \, d_i \tag{14}$$

Where x_i are the input activations, d_i are binary dropout masks (0 with probability $p \; and$ 1 with probability $1 - p$), and p is the dropout rate (probability to remove a neuron). During training, dropout masks (d_i) are applied to input activations, effectively "turning off" a random fraction of neurons. All neurons are employed during inference, but their outputs are scaled by $1 - p$ to account for neurons lost during training.

Introducing dropout makes the model less likely to overfit the training data, forcing the network to acquire more robust features. This regularisation strategy helps to maintain the model's generalisation capabilities, making it more effective when applied to new, previously unknown data. Dropout, used for minimising overfitting, can result in the loss of critical information by randomly deactivating neurons, increasing training complexity and decreasing efficacy in convolutional layers. It also adds processing overhead and is unsuitable for layers with local patterns or spatial hierarchies. Max Pooling overcomes these constraints by lowering feature maps spatially.

3.2.2.5 Max Pooling Layer

After incorporating Dropout into a deep learning model, the process usually proceeds via further layers to consolidate and enhance the collected characteristics for the final prediction stage. Max pooling layers are required in CNNs to reduce the size of feature maps, lowering computing complexity while maintaining salient features and rejecting irrelevant information. The max pooling procedure involves sliding a fixed-size window across the input feature map and picking the highest value within each window, maintaining the most conspicuous characteristics of the data. This method reduces the dimensionality of the input, resulting in smaller feature maps and fewer parameters in later layers. Still, it also adds translational invariance, making the model less susceptible to tiny shifts or distortions in the input. Max pooling, which focuses on the maximum values inside each zone, helps filter out noise and extraneous features, improving the model's robustness and efficiency.

Max pooling on a window of size $m \times n$ with a stride s yields an output feature map F', where each value $f'_{i',j',k}$ represents the maximum value from the corresponding pooling region in F. This process successfully decreases the input dimensions to $H' \times W' \times D'$. The dimensions of the output feature map F' are given by equation (15) (16) (17):

$$H' = \left\lfloor \frac{H-m}{s} \right\rfloor + 1 \quad (15)$$

$$W' = \left\lfloor \frac{W-n}{s} \right\rfloor + 1 \quad (16)$$

$$D' = D \quad (17)$$

Here, H', W', D' are the pooled feature map's height, breadth, and depth. The floor function $\lfloor . \rfloor$ ensures that all dimensions are integers. Max pooling layers minimise dimensionality and preserve features while losing spatial information and simplifying. Fully connected layers address these restrictions by learning complicated non-linear transformations of pooled feature maps, allowing the model to locate features while discarding intermediate information precisely.

3.2.2.6 Fully Connected Layer

The feature mappings in a CNN are usually routed through FC layers after the max pooling layers to refine further and map the retrieved features to a higher-dimensional representation appropriate for the final prediction. This procedure involves flattening the 2D feature maps into a 1D vector and feeding it into densely linked neurons. Unlike convolutional layers, which focus on local spatial patterns, FC layers enable the model to learn global patterns and relationships throughout the feature space.

A model's layers integrate hierarchical representations from previous convolutional and pooling layers, allowing for more complete predictions. They connect neurons from preceding levels and integrate all characteristics. FC layers compensate for the spatial loss of max pooling by aggregating data from all feature maps. FC layers in deep learning models allow models to learn complicated data relationships, adjust for spatial loss, and incorporate hierarchical information to make more accurate predictions, especially in tasks like image classification and stock market prediction.

FC layers help learn complex data correlations but have disadvantages like overfitting and processing costs. Dense layers offer a streamlined method for combining and integrating learnt information into a single prediction, eliminating these restrictions while enhancing model efficiency and scalability.

3.2.2.7 Dense Layer

In a DL model for stock price prediction, utilising a dense layer with a single neuron to generate its outcome prediction is a critical stage that follows the FC layers. This dense layer, the output layer, is responsible for merging the network's learnt hierarchical properties into a single prediction value. The model's predicted value for the future stock price, as determined by convolutional, pooling, and fully connected layers processing of the input data, is represented by this final output in the context of stock price prediction.

A dense layer with a single neuron calculates its output y mathematically as a linear combination of the inputs from the layer before it, usually with the addition of an activation function to add non-linearity. The dense layer's output, y, is determined in equation (18):

$$y = \sigma(w^T x + b) \quad (18)$$

The input vector x from the preceding layer represents the output of the final FC layer. w represents the dense layer's weight vector. The term for bias is b. The activation function, denoted by σ, can be either linear for regression tasks or sigmoid for binary classification. In a deep learning model, combining a dense layer with a single neuron improves interpretability and efficiency and lowers the chance of overfitting. This method uses CPU resources most for real-time applications like stock price forecasting and speeds up inference. This method is essential to contemporary deep learning applications since it increases predicted accuracy and processing efficiency while handling complicated financial data.

3.3 Evaluation Metrics for Stock Prediction

Several essential assessment measures, such as MAE, MSE, RMSE, and R^2, are utilised to assess the effectiveness of the proposed model. Together, these metrics assess how well the model predicts the future by contrasting its predictions with the actual values that have been observed. The MAE provides a simple way to quantify prediction accuracy by calculating the average size of mistakes without considering direction. MSE identifies more enormous mistakes caused by squaring by calculating the average squared disparities between actual and projected numbers. RMSE, derived from MSE, measures the model's prediction spread in the same units as the target variable to improve interpretability. R^2 measures the model's explanatory ability by determining how much of the target variable's variation can be anticipated from the independent variables. When these measures are combined, they provide a comprehensive assessment of the model's predictive power and stock price forecasting accuracy.

4. Result and Discussion

The results and discussion section evaluates the suggested hybrid DL technique for predicting stock prices utilising the Bank Nifty framework. The assessment uses the Bank Nifty dataset, which contains historical data from various banks that provide financial services and covers six essential features for each bank. This dataset is separated into 80% training and 20% testing to ensure a thorough evaluation. The model's performance is assessed using four metrics: MSE, RMSE, MAE, and R^2. Metrics like MSE, RMSE, and MAE are used to gauge how accurate the model is at predicting the future. Lower values indicate close alignment with actual stock prices, while R^2 values approaching 1 indicate a superior fit.

4.1 System and Tool Configuration

The experimental setup and system configuration are illustrated in Table 1.

Table 1. Experimental setup and system configuration

Tool	PYTHON 3
OS	Windows 7 (64-bit)
RAM	8 GB RAM
Processor	Intel Premium

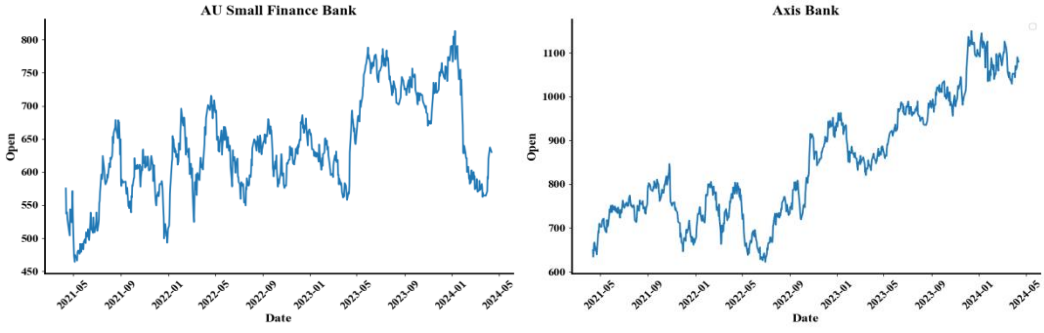
4.2 Dataset Description

An acceptable dataset is essential for creating a stock market price prediction model. From the available Bank Nifty Stock Price Prediction dataset, stock market prices were predicted using the proposed DL technique. The Bank Nifty Stock Price Prediction dataset is an extensive compilation of financial data that is concentrated on the Bank Nifty index. The index represents the performance of the largest and most liquid Indian banking firms. This dataset typically includes daily or minute-by-minute stock prices, as well as other relevant financial data such as opening and closing prices, high and low prices, share volume, and potentially additional technical indicators such as moving averages, RSI (Relative Strength Index), and MACD (Moving Average Convergence Divergence).

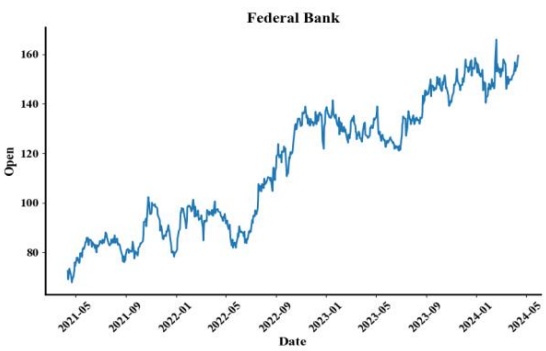
The system's ability to predict time series for every bank included in the Bank Nifty was tested on an actual network. As seen in Fig.4, the Bank Nifty is based on historical data from several banks that provide financial services. The Bank Nifty dataset has six characteristics for each of these, which are listed in Table 2. Eighty percent of the dataset was used for testing, compared to only twenty percent for training.

Table 2. Description of Bank Nifty Attributes

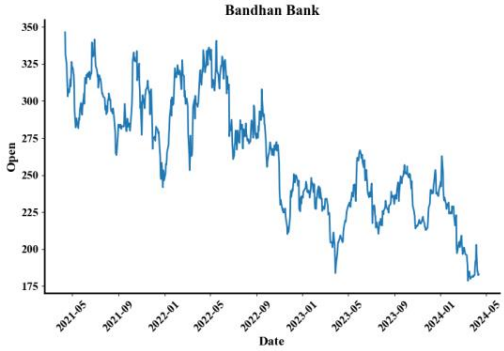
Attributes	**Description**
Open	Stock price on a trading day when the market begins.
Close	The stock price at the end of a trading day.
High	The stock's highest value on that specific trading day.
Low	The stock's lowest price on that specific trading day.
Adjusted Close	After accounting for any business actions, the closing price will indicate stock value.
Volume	Shares traded on a trading day.



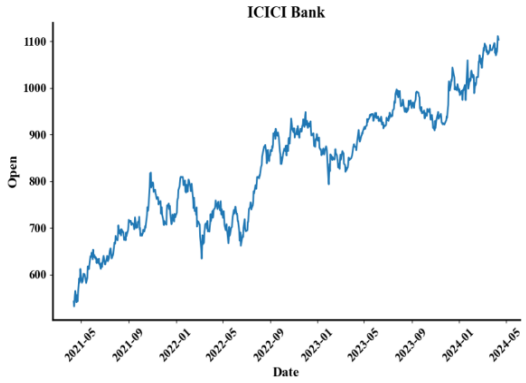
(a) AU Small Finance Bank (b) Axis Bank



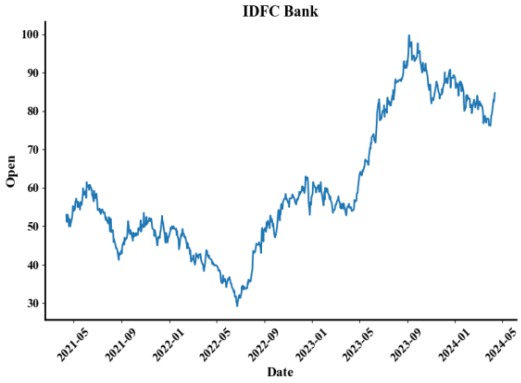
(c) Federal Bank



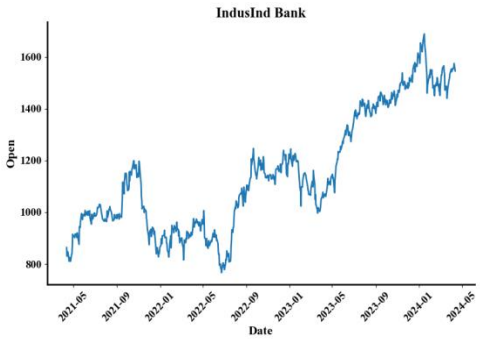
(d) Bandhan Bank



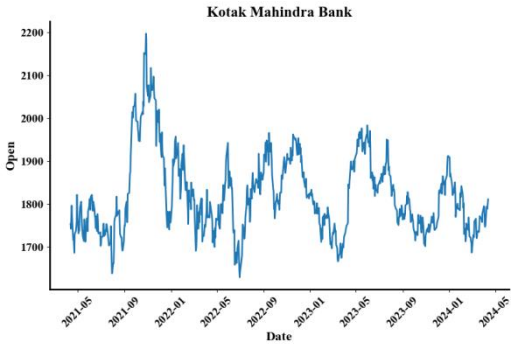
(d) ICICI Bank



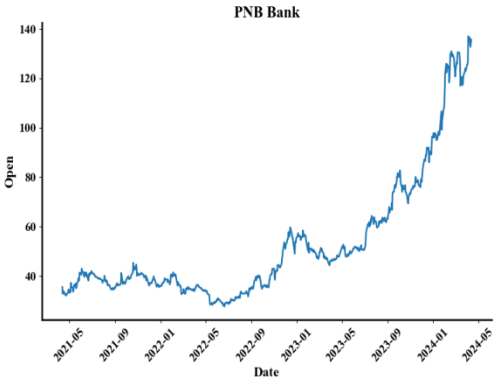
(e) IDFC Bank



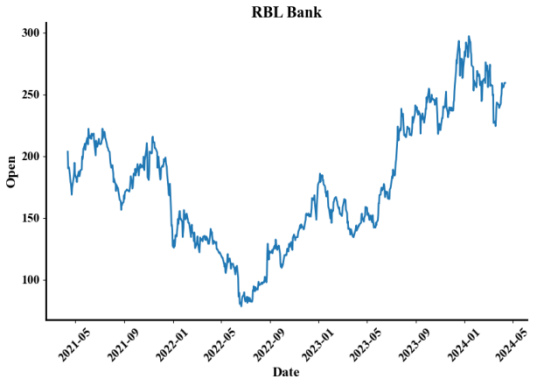
(f) IndusInd Bank



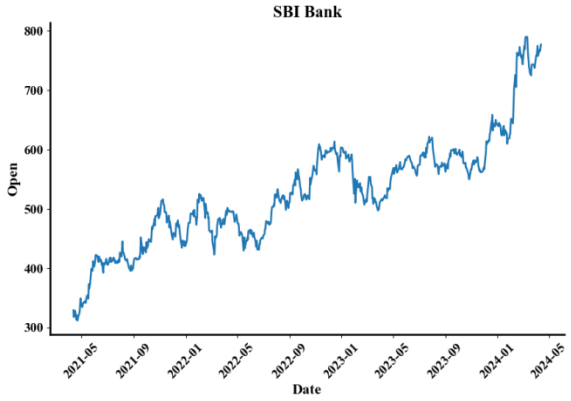
(g) Kotak Mahindra Bank



(h) PNB Bank



(i) Kotak Mahindra Bank



(j) SBI Bank

Fig. 4. Historical Data for all the banks under Bank Nifty (NSE BANK)

4.3 Evaluation of Performance Metrics

The system used the MSE, RMSE, MAE, and R^2 to assess the results and the established forecast approach. The projected value will be closer to the true value if the MSE, RMSE, and MAE are smaller. Similarly, the closer the coefficient R^2 is to 1, the better the model matches the data. The calculation of MSE, MAE, RMSE, R^2 is given in equation (19) to (22):

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{Y}_i - Y_i\right), \tag{19}$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|\left(\hat{Y}_i - Y_i\right)\right|, \tag{20}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\hat{Y}_i - Y_i\right)}, \tag{21}$$

$$R^2 = \sum_{i=1}^{N}\left(\hat{Y}_i - \bar{Y}\right)^2 / \sum_{i=1}^{N}(Y_i - \bar{Y})^2, \tag{22}$$

Where N stands for the sample size, Y_i is the real value,$\hat{Y}_i$ is the model prediction value, and $\bar{Y}$ is the mean value of Y_i.

Four key metrics—RMSE, MSE, R^2 and MAE—are used to evaluate the proposed technique for several Nifty Banks. These loss functions are used to assess the model's performance for eleven Nifty Banks, as indicated in Table 3, offering an extensive analysis of the model's efficacy and accuracy. The model's ability to predict stock prices across a wide range of financial institutions is demonstrated by Fig.5-10, which shows the MSE and MAE values for each of the eleven banks—Kotak, Axis, Federal, ICICI, AU Small Finance, IDFC, Bandhan, IndusInd, Punjab, RBI, and SBI—among them.

Table 3. Performance Metrics for the Proposed Model

BANKS	MSE	RMSE	MAE	R^2
AU FIN	0.203	0.451	0.272	159.680
Axis	0.080	0.283	0.167	4.374
SBI	0.071	0.267	0.150	2.050
RBI	0.082	0.287	0.172	1.330
PNB	0.050	0.224	0.105	1.937
KOTAK	0.244	0.494	0.277	3.316
INDUS IND	0.078	0.280	0.165	3.088
IDFC	0.086	0.293	0.162	1.860
ICICI	0.068	0.261	0.158	4.402
BANDHAN	0.127	0.356	0.208	1.261
FEDERAL	0.086	0.293	0.164	10.060

A hybrid deep learning model for predicting stock prices across many Banks is presented with its performance measured using R-squared (R^2), Mean Absolute Error (MAE), Root Mean Square Error (RMSE) and Mean Square Error (MSE). With PNB obtaining the lowest MSE (0.050) and RMSE (0.224), indicating it has the most accurate forecasts among the listed banks, lower MSE and RMSE values indicate higher prediction accuracy. The MAE values represent the average absolute errors, and PNB once more exhibits the best performance (0.105). The amount of variance the model explains, or R-squared, varies significantly. AU FIN has an exceptionally high R^2 (159.680), which may indicate an anomaly or miscalculation. Other banks with predictive solid power include Federal (100.060) and ICICI (4.402). The model works well overall, but its accuracy varies significantly amongst banks, as shown by the various performance measures. This emphasises the need for model validation and possible customisation for individual stocks to maximise prediction accuracy.

The model's inconsistent performance across banks can be explained by several variables, such as each bank's distinct characteristics and stock price volatility, the availability and quality of historical data, and the unique financial health and positioning of each bank in the market. It may be easier to accurately simulate banks with lower MSE and RMSE values, such as SBI and ICICI, because their stock price changes are more steady and predictable. On the other hand, banks with more excellent error metrics, such as AU FIN and KOTAK, might show less consistent or more erratic patterns in their stock prices. The model's performance can also be impacted by variations in market capitalisation, trading volume, and investor behaviour for the stocks of each bank. Variations in the financial statements, the effect of news, and the general situation of the economy that affects each bank differently could all add to the discrepancies in the prediction's accuracy.

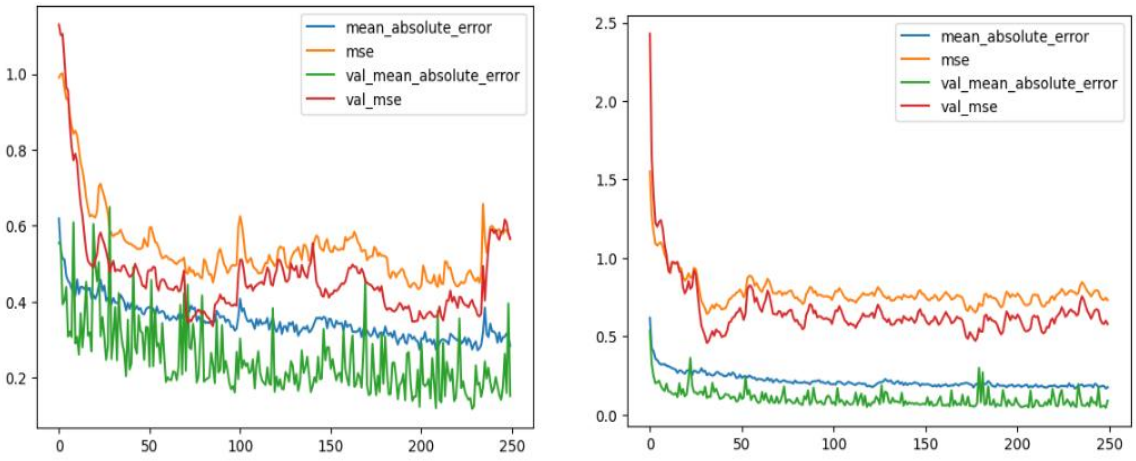


Fig. 5. Performance of MAE and MSE Loss of training and testing data for (a) AU Finance Bank (b) Axis Bank

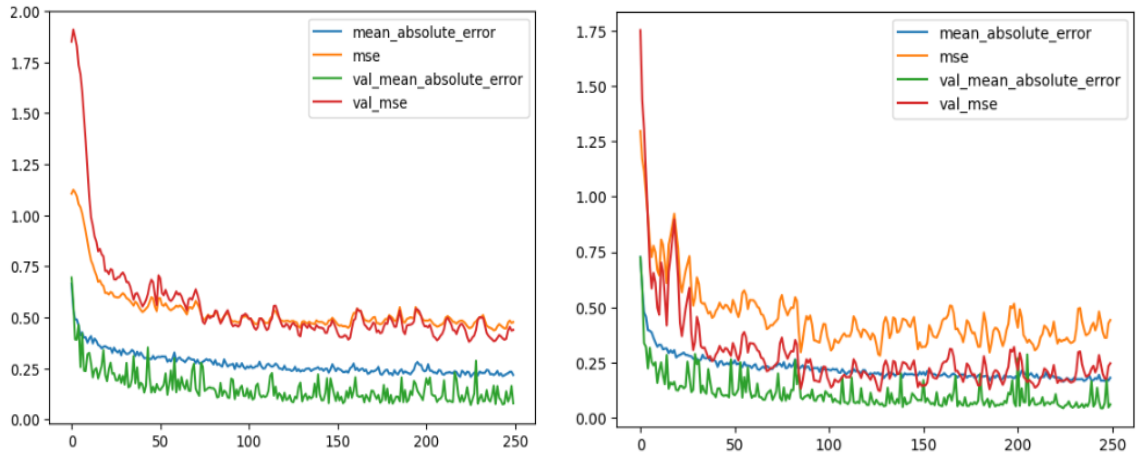


Fig. 6. Performance of MAE and MSE Loss of training and testing data for (a) Bhandhan Bank (b) Federal Bank

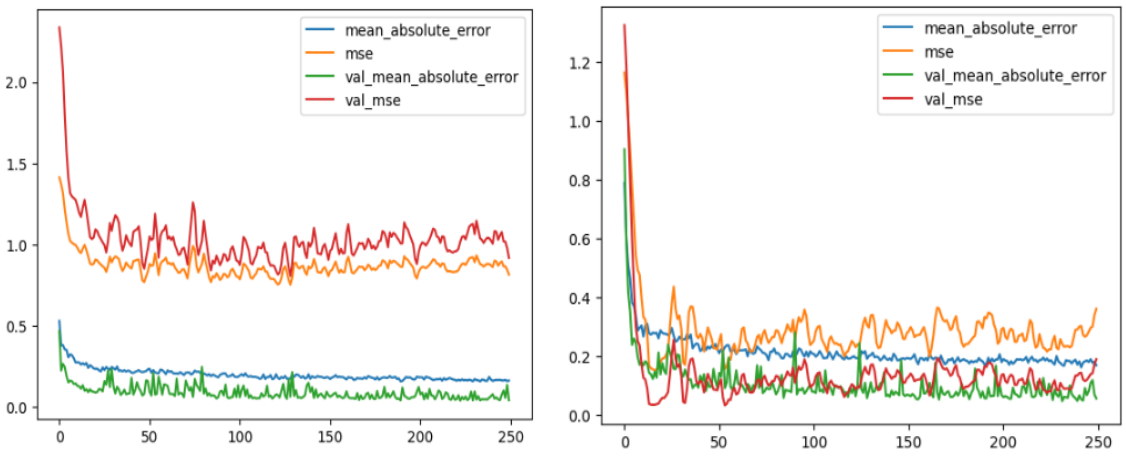


Fig. 7. Performance of MAE and MSE Loss of training and testing data for (a) ICICI Bank (b) IDFC Bank

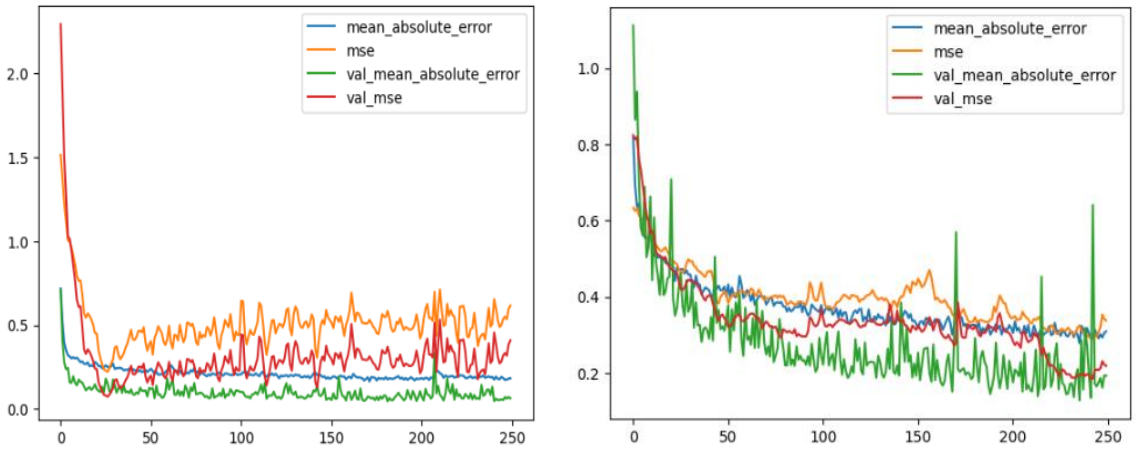


Fig. 8. Performance of MAE and MSE Loss of training and testing data for (a) IndusInd Bank (b) Kotak Bank

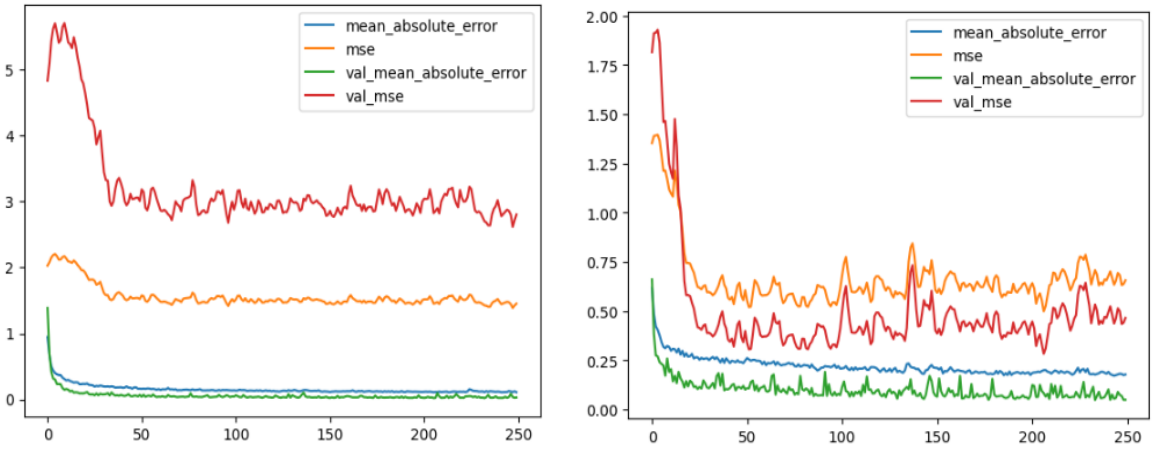


Fig. 9. Performance of MAE and MSE Loss of training and testing data for (a) PNB Bank (b) RBL Bank

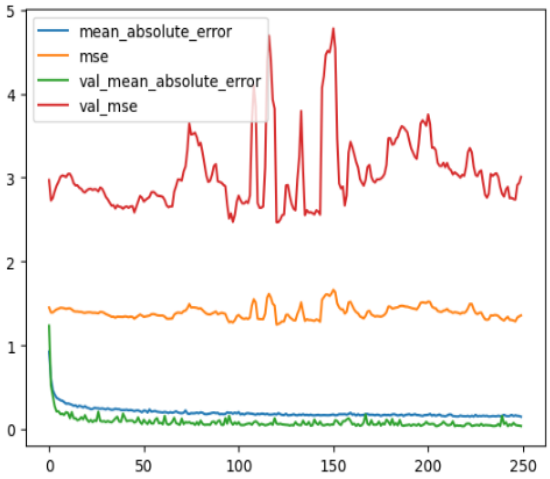


Fig. 10. Training and Validation Loss Curves for MAE and MSE.

Fig.10. Performance of MAE and MSE Loss of training and testing data for SBI Bank PNB had the lowest average squared and root squared discrepancies (RMSE and MSE, respectively) between the expected and actual values among the evaluated banks, at 0.050 and 0.224, respectively. This implies that PNB's projected stock price aligns more with the actual values. On the other hand, Kotak Bank displays the highest RMSE of 0.494 and MSE of 0.244, indicating higher prediction mistakes and variability. PNB also has the lowest MAE value, 0.105, indicating the most petite average magnitude errors. With the most excellent R^2 score of 10.060, Federal Bank shows that the model can explain a sizable amount of the variance in its stock price movements. In contrast, Axis Bank exhibits a comparatively lower explanatory power (R^2 = 4.374), although still substantial. These indicators, taken together, demonstrate the model's uneven efficacy across banks, emphasising some banks' strengths in prediction accuracy while pointing out areas in which other banks may benefit from improvement. These findings are critical to improving the hybrid deep learning technique for improved stock price forecasting.

4.4 Comparison Analysis

The proposed stock prediction method's mathematical foundation is defined by equations (20) and (21). MAE and RMSE were used to assess the process. Next, the effectiveness of this proposed method was contrasted with that of other models that are currently in use, including PRE_DNN, DNN, Stacked Bi-LSTM E-optimized CNN and ANN, all of which have been used to forecast stock prices using the NIFTY bank stock price dataset. A comparison of the MAE and RMSE values for four banks between the suggested model and the previously described current models is shown in Table 4 and Fig. 11 to 14. Applied to real-world financial statistics, this comparison demonstrates how practical the recommended approach is in increasing forecast accuracy and decreasing error measures.

Table 4. Comparison of models based on MAE and RMSE Loss

Technique	KOTAK		ICICI		AXIS		SBI	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
PRE_DNN [3]	9.50	6.50	10.30	5.60	9.90	7.10	8.53	6.30
DNN [11]	13.50	11.50	14.50	13.60	13.30	12.60	13.50	11.20
ANN[10]	-	-	15.12	19.94	-	-	17.43	23.15
Stacked Bi-LSTM E-optimized CNN [22]	2.29	1.763	3.04	2.31	2.35	0.81	2.76	2.10
Proposed model	0.28	0.494	0.15	0.34	0.17	0.28	0.13	0.24

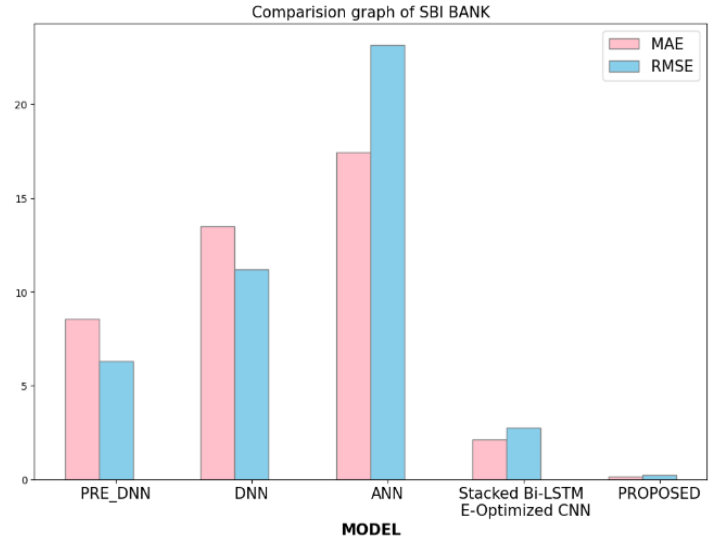


Fig. 11. Comparison of various model for SBI Bank

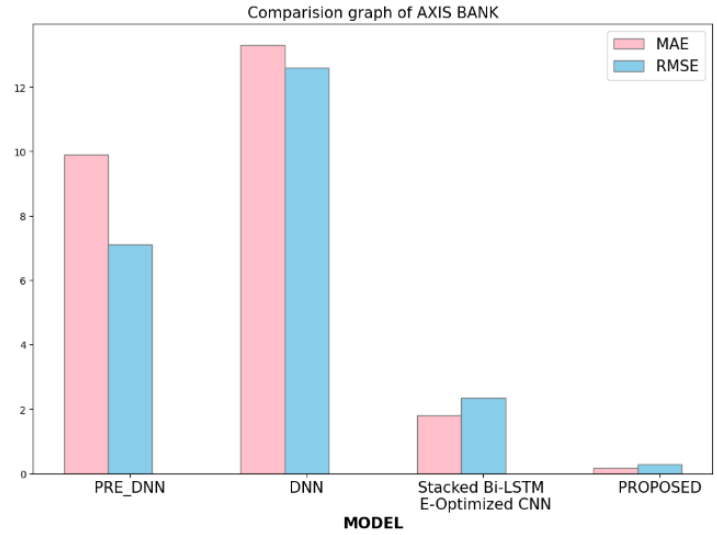


Fig. 12. Comparison of various models for Axis Bank

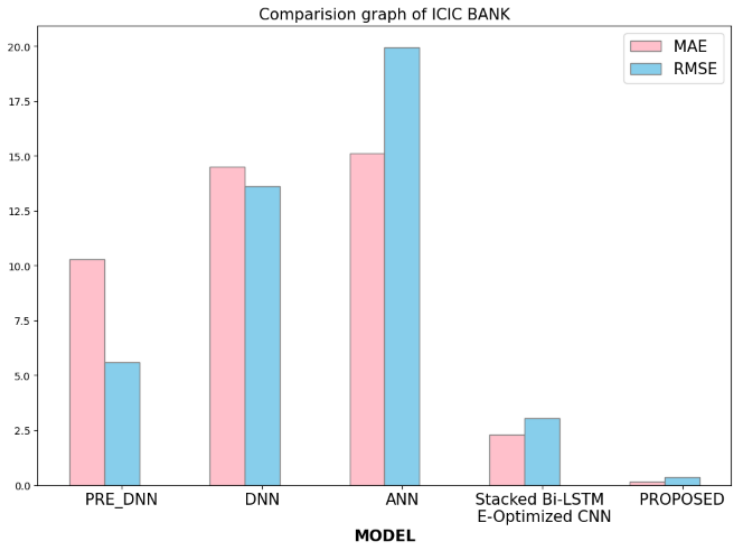


Fig. 13. Comparison of various models for ICICI Bank

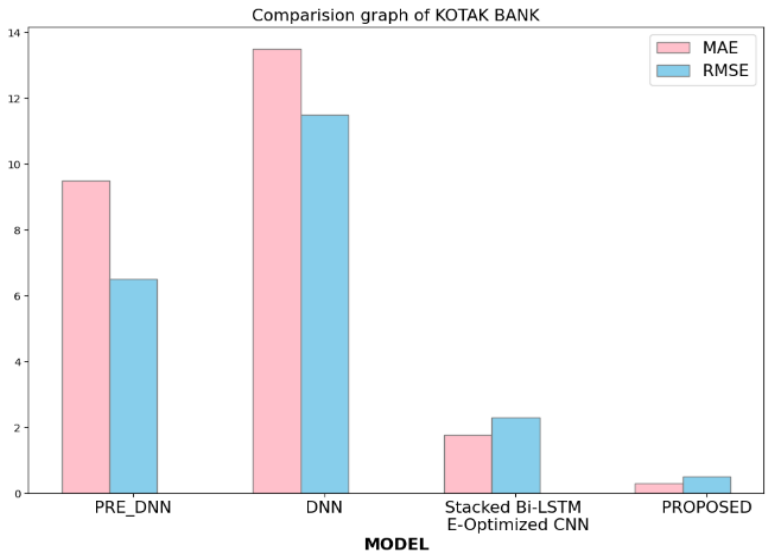


Fig. 14. Comparison of various models for Kotak Bank

A comparison investigation of stock price prediction models from four banks—Kotak, ICICI, Axis, and SBI—using measures such as MAE and RMSE indicates considerable performance inequalities. The suggested approach is more accurate, with lower errors when predicting stock prices: Kotak MAE 0.28 and RMSE 0.494, ICICI MAE 0.15 and RMSE 0.34, Axis MAE 0.17 and RMSE 0.28, and SBI MAE 0.13 and RMSE 0.24. On the other hand, conventional models like DNN and ANN display significantly greater error rates. DNN has MAEs between 13.30 and 14.50 and RMSEs between 11.20 and 13.60, and ANN displays significant mistakes for SBI (MAE: 17.43, RMSE: 23.15) and ICICI (MAE: 15.12, RMSE: 19.94). While the PRE_DNN model outperforms DNN and ANN, it still falls behind the suggested model, which has more significant bank errors. The recommended method outperforms the Stacked Bi-LSTM E-optimized CNN model, which performs comparatively well. The proposed model's capacity to predict stock prices more reliably and accurately is demonstrated by the notable declines in MAE and RMSE observed across various financial organisations.

5. Conclusion

The suggested hybrid DL technique significantly improves stock price prediction by adding DNN components specialised for time series analysis. While the Transformed Moving Average Fast-RNN Hybrid with Advanced CNNs addresses computational issues and enables practical training and inference, pre-processing with the Moving Average Z-Transformation effectively emphasises long-term patterns and filters out noise. The model's Enhanced Deep-CNN layer captures the nonlinear and volatile nature of stock prices, which combines convolutional layers, batch normalisation, Leaky ReLU, dropout, and max pooling. This produces fully connected layers that employ a dense layer to turn features into precise predictions. The better prediction accuracy of the model is validated by performance metrics including R^2, RMSE, MAE, and MSE, and it demonstrates superior prediction accuracy with MAEs of 0.28, 0.15, 0.34, 0.17, and 0.13 for Kotak, ICICI, Axis, and SBI, respectively, outperforming previous models. Since it more accurately predicts long-term trends and reduces the complex patterns in the data than traditional models, the suggested hybrid deep learning model is utilised for predicting stock prices. The Moving Average Z-Transformation, Fast-RNN, and Enhanced Deep-CNN combo uses cutting-edge computational techniques to handle the nonlinearity and complexity of stock price data. The benefit of this research is that prediction accuracy is increased through the hybrid deep learning approach's ability to capture long-term trends and reduce short-term fluctuations. Although the proposed model attained good results, it has limitations, such as the complicated structure of the model, which increases the risk of overfitting; therefore, careful tuning and validation are required. Future research might concentrate on increasing computational efficiency and adding more financial variables to further improve model performance.

References

[1] G. Sonkavde, D. S. Dharrao, A. M. Bongale, S. T. Deokate, D. Doreswamy, & S. K. Bhat, “Forecasting stock market prices using machine learning and deep learning models: A systematic review, performance analysis and discussion of implications,” International Journal of Financial Studies, vol. 11, no. 3, pp. 94, 2023.

[2] Obthong, Mehtabhorn, Nongnuch Tantisantiwong, Watthanasak Jeamwatthanachai, and Gary Wills, “A Survey on Machine Learning for Stock Price Prediction: Algorithms and Techniques. Paper presented at 2nd International Conference on Finance, Economics, Management and IT Business, Prague,” Czech Republic, May, pp. 5–6, 2020.

[3] Nikou, Mahla, Gholamreza Mansourfar, and Jamshid Bagherzadeh, "Stock price prediction using DEEP learning algorithm and its comparison with machine learning algorithm," Intelligent Systems in Accounting, Finance, and Management, vol. 26, no. 4, pp. 164-174, 2019.
[4] Yang, Can, Junjie Zhai, and Guihua Tao, "Deep learning for price movement prediction using convolutional neural network and long short-term memory," Mathematical Problems in Engineering, 2020.
[5] G. Ji, J. Yu, K. Hu, J. Xie, & X. Ji, "An adaptive feature selection schema using improved technical indicators for predicting stock price movements," Expert Systems with Applications, vol. 200, pp. 116941, 2022.
[6] Jiang, Weiwei, "Applications of deep learning in stock market prediction: recent progress," Expert Systems with Applications, vol. 184, pp. 115537, 2021.
[7] Nabipour, Mojtaba, Pooyan Nayyeri, Hamed Jabani, S. Shahab, and Amir Mosavi, "Predicting stock market trends using machine learning and deep learning algorithms via continuous and binary data; a comparative analysis," IEEE Access, vol. 8, pp. 150199-150212, 2020.
[8] Mehtab, Sidra, and Jaydip Sen, "Stock price prediction using convolutional neural networks on a multivariate time-series," arXiv preprint arXiv: 2001.09769, 2020.
[9] K. C. Tseng, O. Kwon, & L. C. Tjing, "Time series and neural network forecasts of daily stock prices," Investment Management and Financial Innovations, vol. 9, no. 1, pp. 32-54, 2012.
[10] D. I. Ajiga, R. A. Adeleye, T. S. Tubokirifuruar, B. G. Bello, N. L. Ndubuisi, O. F. Asuzu, & O. R. Owolabi, "Machine learning for stock market forecasting: a review of models and accuracy," Finance & Accounting Research Journal, vol. 6, no. 2, pp. 112-124, 2024.
[11] T. Choudhury, & S. Yadav, "Predicting Stock Price Using Support Vector Machine," International Journal of Business and Information Technology, 2019.
[12] S. Gu, B. Kelly, & D. Xiu, "Empirical Asset Pricing via Machine Learning," The Review of Financial Studies, vol. 33, no. 5, pp. 2223-2273, 2020.
[13] O.B. Sezer, M.U. Gudelek, & A.M. Ozbayoglu, "Financial time series forecasting with deep learning: A systematic literature review," Applied soft computing, vol. 90, pp. 106181, 2020.
[14] X. Ji, J. Wang, & Z. Yan, "A stock price prediction method based on deep learning technology," International Journal of Crowd Science, vol. 5, no. 1, pp. 55-72, 2021.
[15] Kumar, Krishna, and Md Tanwir Uddin Haider, "Enhanced prediction of intra-day stock market using metaheuristic optimization on RNN–LSTM network," New Generation Computing, vol. 39, pp. 231-272, 2021.
[16] A. K. Mahapatra, N. Panda, & B. K. Pattanayak, "Hybrid PSO (SGPSO) with the Incorporation of discretization operator for training RBF neural network and optimal feature selection," Arabian Journal for Science and Engineering, vol. 48, no. 8, pp. 9991-10019.
[17] Hashim, A. Fatma, Kashif Hussain, H. Essam Houssein, S. Mai Mabrouk, and Walid Al-Atabany, "Archimedes optimization algorithm: a new metaheuristic algorithm for solving optimization problems," Applied Intelligence, vol. 51, pp. 1531-1551, 2021.
[18] Faramarzi, Afshin, Mohammad Heidarinejad, Brent Stephens, and Seyedali Mirjalili, "Equilibrium optimizer: A novel optimization algorithm," Knowledge-Based Systems, vol. 191, pp. 105190, 2020.
[19] Liang, Mengxia, Shaocong Wu, Xiaolong Wang, and Qingcai Chen, "A stock time series forecasting approach incorporating candlestick patterns and sequence similarity," Expert Systems with Applications, vol. 205, pp. 117595, 2022.
[20] D. Wu, X. Wang, & S. Wu, "A hybrid method based on extreme learning machine and wavelet transform denoising for stock prediction," Entropy, vol. 23, no. 4, pp. 440, 2021.
[21] Tao, Meiyao, Shanshan Gao, Deqian Mao, and Hong Huang, "Knowledge graph and deep learning combined with a stock price prediction network focusing on related stocks and mutation points," Journal of King Saud University-Computer and Information Sciences, vol. 34, no. 7, pp. 4322-4334, 2022.
[22] Wu, Dingming, Xiaolong Wang, and Shaocong Wu, "A hybrid framework based on extreme learning machine, discrete wavelet transform, and autoencoder with feature penalty for stock prediction," Expert Systems with Applications, vol. 207, pp. 118006, 2022.
[23] N. N. Vo, X. He, S. Liu, & G. Xu, "Deep learning for decision making and the optimization of socially responsible investments and portfolio," Decision Support Systems, vol. 124, pp. 113097, 2019.
[24] Kanwal, Anika, Man Fai Lau, P.H. Sebastian Ng, Kwan Yong Sim, and Siva Chandrasekaran, "BiCuDNNLSTM-1dCNN— A hybrid deep learning-based predictive model for stock price prediction," Expert Systems with Applications, vol. 202, pp. 117123, 2022.

Authors' Profiles

Swarnalata Rath is currently working as an Assistant Professor in the Department of Computer Science & Engineering, NMIET, Bhubaneswar, Odisha. She is currently pursuing Ph.D, in Computer Science &Engineering at Siksha 'O' Anusandhan(Deemed to be University), Odisha. She has published 2 International Journal papers, 2 Conference papers and 1 patents. She has attended various Seminars, Conferences and Workshops at National level and International level.

Nilima R. Das is currently working as an Associate Professor in the Department of Computer Application, Siksha 'O' Anusandhan (Deemed to be University), Odisha. She has published 10 National and International Journal papers, 10 Conference papers and 3 patents. She has attended various Seminars, Conferences and Workshops at National level and International level.

Binod Kumar Pattanayak completed his M. S. in Computer Engineering in the year 1992 from NTU Kharkov Polytechnical Institute, Kharkov, USSR, Ph. D. in Computer Science and Engineering in the year 2011 from Siksha 'O' Anusandhan University, Bhubaneswar, India. He is currently working as a Professor in the Department of Computer Science and Engineering, Institute of Technical Education and Research, Siksha 'O' Anusandhan Deemed to be University, Bhubaneswar, India. His research interests include Internet of Things (IoT), Artificial Intelligence, Big Data Analytics and Cloud Computing. There are 235 research publications in journals and conferences of international repute to his credit. As many as 21 research scholars have been awarded with Ph. D. degree and 9 scholars are continuing their Ph. D. research work under his supervision. He has visited Build Bright University, Phnompenh, Cambodia and Universite des Mascreignes, Mauritius as a visiting professor. He belongs to the editorial boards of various reputed peer-reviewed international journals. He has edited one Springer Book. He has acted as General Chair, Program chair in various international conferences.

How to cite this paper: Swarnalata Rath, Nilima R. Das, Binod Kumar Pattanayak, "Next-Gen Market Predictor: Transformed Moving Average Fast-RNN Hybrid with Advanced CNNS", International Journal of Image, Graphics and Signal Processing(IJIGSP), Vol.17, No.3, pp. 104-122, 2025. DOI:10.5815/ijigsp.2025.03.06