Received: 29 October 2020 | Revised: 17 February 2021 | Accepted: 18 May 2021

DOI: 10.1049/cit2.12052

ORIGINAL RESEARCH

Stock values predictions using deep learning based hybrid models

Konark Yadav[1] | Milind Yadav[2] | Sandeep Saini[1]

[1]Department of Electronics and Communication Engineering, The LNM Institute of Information Technology, Jaipur, India

[2]Department of Computer Science and Engineering, Rajasthan Technical University, Kota, Rajasthan, India

Correspondence
Sandeep Saini, Department of Electronics and Communication Engineering, The LNM Institute of Information Technology, Jaipur, India.
Email: sandeep.saini@lnmiit.ac.in

Abstract
Predicting the correct values of stock prices in fast fluctuating high-frequency financial data is always a challenging task. A deep learning-based model for live predictions of stock values is aimed to be developed here. The authors' have proposed two models for different applications. The first one is based on Fast Recurrent Neural Networks (Fast RNNs). This model is used for stock price predictions for the first time in this work. The second model is a hybrid deep learning model developed by utilising the best features of FastRNNs, Convolutional Neural Networks, and Bi-Directional Long Short Term Memory models to predict abrupt changes in the stock prices of a company. The 1-min time interval stock data of four companies for a period of one and three days is considered. Along with the lower Root Mean Squared Error (RMSE), the proposed models have low computational complexity as well, so that they can also be used for live predictions. The models' performance is measured by the RMSE along with computation time. The model outperforms Auto Regressive Integrated Moving Average, FBProphet, LSTM, and other proposed hybrid models on both RMSE and computation time for live predictions of stock values.

1 | INTRODUCTION

Economies around the globe are digital now and dependent on each other. A multinational company is having its impact on a lot of companies. The fluctuations in the stock values of such companies can change the economic scenarios for multiple stakeholders. Thus, forecasting the stock values is becoming more crucial now. Forecasting is the process of predicting the future value of any series by considering the previous patterns or long historical data. For example, if the price of gold is increasing every year at Christmas time, then we can predict a similar trend for the current year as well and plan the purchase well in advance to avoid the high rates at Christmas time. Similarly, computational models can help us in predicting the weather for the next day, week, or month as well. With the high volume of money involved, the stock market values have attracted the attention of computer scientists as well to design models and architectures for precise stock value prediction. A lot of such systems have been developed with high accuracies during the past decades as well [1]. Forecasting problems can be further divided into three categories and listed as follows:

a. Short-term forecasting, where prediction happens for the next second, minute, hour, day, or month
b. Medium-term forecasting, where prediction happens for the next 1 year, or 2 years.
c. Long-term forecasting, where prediction happens beyond 2 years.

1.1 | Motivation

Stocks are not a simple time series with only one factor affecting the outcome. These can either be univariate or multivariate. Univariate stocks are rare and they are dependent on only one factor or only one company's performance. With the emerging partnerships and dependence of every big company on its partner's stocks, the second type of stock, that is, multivariate is more common now. So, in such cases, the prediction of exact future stock values can help a lot of investors and stakeholders. This is the motivation behind our proposed model.

1.2 | Stock market prediction

The first model to predict the outcome of a time series was first proposed by Ahmad and Cook [2] in 1979. In this work, Auto Regressive Integrated Moving Average (ARIMA) model was introduced, which is one of the most trusted models for time series forecasting even now. This is also a reason behind the fact that a lot of conventional models are based on Auto Regression (AR) and Moving Average (MA) and Exponential Smoothing [3, 4] and generalized autoregressive conditional heteroskedasticity models [5]. The existing stock market prediction methods can be classified as follows.

1. Fundamental Analysis is concerned with the company that underlies the stock itself. In such methods, the historical performance, as well as the credibility of the company's accounts, is evaluated to predict future stock values. These methods are stochastic and have limited options to consider unprecedented events to predict the stock in such cases.
2. Technical Analysis. These methods are not considering the working principles of the company but mostly consider the trends and historical stock values. AR, MA, ARIMA, and similar methods are popular methods in this category. These methods are more effective for short-term predictions rather than long-term predictions.
3. Machine Learning (Time Series Forecasting). Artificial Intelligence is emerging as a robust solution for time series forecasting. These methods include artificial neural networks (ANNs) and Genetic Algorithms (GAs). The most common form of ANN in use for stock market prediction is the feed-forward network utilising the backward propagation of errors algorithm to update the network weights. In recent years, Recurrent Neural Network (RNN) has become the best ANN for time series forecasting.

We have focussed on the Deep Learning-based approaches in this work. We have studied the latest models and architectures proposed explicitly for stock market prediction. After considering the shortcomings in terms of their computational time and the Root Mean Squared Error (RMSE) values, we have proposed two different models in this work. We have focussed on two performance parameters, that is, execution time and RMSE. A model can be very accurate in prediction but very slow in computing the output. Such a model cannot be applied for live stock value prediction but very useful for long-term predictions. On the other hand, a faster model with acceptable RMSE can be utilised for live predictions as well. The first model is based on FastRNN architecture. This model is designed to provide faster and accurate results. The second model is proposed by hybridizing FastRNN with Convolutional Neural Network (CNN) and Bi-Directional Long Short Term Memory (Bi-LSTM) networks. This model not only provides the output with less execution time but also with improved RMSE values.

1.3 | Organisation

The rest of the paper is organised as follows. We have provided the literature review of the existing machine learning and deep learning-based models for stock market prediction in section 2. We have focussed on single network-based as well as hybrid models in this section. The proposed models are described in complete detail in section 3. We have explained the mathematical model as well as structural augmentations made to implement these models in the same. Section 4 consists of details of our experimental setup, datasets, and simulation results for short-term and long-term stock price prediction. We conclude our work in section 5.

2 | LITERATURE REVIEW

We have described that there are three major methods for stock market prediction. We have focussed only on machine learning and more specifically on deep learning-based approaches in our literature review. One of the earliest applications of machine learning models for stock market forecasting was from Tay and Cao [6] in 2001. They used Support Vector Machine (SVM) to verify the feasibility of SVMs for financial series forecasting. The promising results motivated a lot of research groups to explore more machine learning algorithms for the same. In 2003, Egeli et al. [7] proposed a Multi-Layer Perceptron and Generalised Feed Forward network-based architecture for financial forecasting. These models were trained and tested on MAs for 5 and 10-day periods. The results outperformed conventional ARIMA-based approaches. In 2005, Enke and Thawornwong [8] combined conventional data mining algorithms with neural networks to predict stock values. The authors used the Probabilistic Neural Network in their model that was having three feed-forward layers. In the same year, Huang et al. used an SVM-based model to predict the stock values and showed that the SVM-based model outperforms Linear Discriminant Analysis, Quadratic Discriminant Analysis, and Elman Backpropagation Neural Networks.

With the encouraging results from single network-based models for time series forecasting, hybrid models were also explored for the purpose. In 2001, Abraham et al. [9] proposed one of the first hybrid models that were based on neuro-fuzzy logic and ANN. In 2005, Armano et al. [10] proposed the model with hybridisation of a genetic classifier designed to control the activation of a feedforward ANN for performing a locally scoped forecasting activity. In 2007, Fu et al. [11] analysed the problem based on a Fuzzy Cerebellar Model Articulation Controller – Bayesian Ying Yang neural network. This proposed model was motivated by the Chinese ancient Ying-Yang philosophy that states that everything in the universe can be viewed as a product of constant conflict between opposites, Ying, and Yang. In 2008, Choudhary and Garg [12] proposed the GA-SVM model for stock market prediction. The model is a hybrid version of the GA and SVMs. The system was tested on three of India's biggest companies, that is, TCS, Infosys, and RIL that had a trading data of 1386 days from 2002 to 2008. Further, it was tested on the data of 30 companies. The hybrid model outperformed the SVM-based models in terms of RMSE. In 2009, Tsai and Wang [13] also performed the stock price forecasting by using a hybrid model designed using a decision tree (DT) and ANNs. The proposed model, DT-ANN, had an accuracy of 77% and was among the most precise systems at that time.

In this decade, the focus has shifted to deep learning-based models. Ding et al. [14] developed an event-driven deep learning model. In this model, a variant of CNN, that is, Deep CNN was used to predict the stock values. The events are extracted from the news articles and stored as dense vectors, trained using a novel neural tensor network. This dense network was trained using the Deep-CNN. Akita et al. [15] applied deep learning models Paragraph Vector, and Long Short-Term Memory (LSTM) to financial time series forecasting. The model also utilised the news article data and converted those into the Paragraph vector, which was then fed to LSTM to predict the stock prices. This model was tested on the Tokyo Stock Exchange. Fischer and Krauss [16] also proposed a similar LSTM-based architecture. The model outperformed random forest, a deep neural network (DNN), and LOG-based models. CNN has been employed by several researchers for the problem. Hoseinzade and Haratizadeh [17] proposed CNNpred, which was a CNN-based model for establishing the relationships between different stock markets across the globe and showed the positive correlation between the trends across the global stock exchanges. Gudelek et al. [18] enhanced the regular CNN model to the two-dimensional model and employed it for the task. The first convolutional layer had 32 filters of size 28×28 and the second one had 64 filters of the same sizes. This improved the RMSE values. Eapen et al. [19] presented a hybrid model that was made using CNN and Bi-LSTM. The proposed model was 9% better than a single network-based model. Selvin et al. [20] had proposed a similar architecture that was based on the hybridisation of LSTM, RNN, and CNN. In the next section, we explain our proposed models which aim to improve the RMSE values as well as provide faster results.

Significant work was carried out on stock by Nabipour et al. in the field of stock market predictions. The authors have carried out Stock Market predictions Using Machine Learning and Deep Learning Algorithms via Continuous and Binary Data [21, 22]. Ecer et al. have used GAs and Particle Swarm Optimisation for Modelling Stock Price Index Prediction [23, 24]. These techniques can be used in various other domains as well [25, 26].

3 | PROPOSED MODEL

In designing a stock market price prediction model, the foremost requirement is the availability of a suitable dataset. We have considered four companies for our study and those are Facebook Inc., Uber Inc., Apple Inc., and Nike Inc. from the New York Stock Exchange. We have obtained the stock values from Yahoo finance.[1] The dataset includes information about day stamp, time stamp, transaction ID, the stock price (open and close), and volume of stock sold in each minute interval. For our model, we have used the close price for each stock. Our work also aims on creating a prototype for live prediction. We consider a working duration of 8 h and divide those hours into training and testing time. We are predicting the future prices of each minute for the next 50 min by keeping the initial 7 h 10 min data in training. The best window length was identified by calculating the root mean square error for various window sizes.

We kept the size of the data the same for all the stocks, that is, each stock has 430 rows and the model was trained on 40 epochs. We have considered the error and computation time of each model for our study. If the loss (mean squared error) for the current epoch is less than the value obtained from the previous epoch, the weight matrix for that particular epoch is stored. After the completion of the training process, each of these models was tested on the remaining 50 values. In this process, the model with the least RMSE is taken as the final model for prediction.

To compare our model with the existing similar models, we have initially considered a few baseline models and then the state-of-the-art models. We have considered the following baseline models for our study.

1. ARIMA is a widely known model for time series forecasting that works on the concept of MA. It is based on the trend stationarity, and seasonality of the data. Here, we have to first make the input data stationary and then after finding their auto-correlation values and partial auto-correlation, we would be able to forecast. The only glitch with this model is that it cannot be automated for all kinds of stocks, as we have to set different p, d, q (number of auto-regressive terms, number of non-seasonal differences needed for stationarity, number of lagged forecast errors) values for different stocks.
2. LSTM is a well-established model used for time series forecasting and frequently applied for stock market price prediction [27–29]. It is a derived RNN with forget gate functionality. LSTMs are very effective for long as well as short-term predictions.
3. FBProphet: is another time series forecasting model introduced by Facebook Inc [30]. This model required very less computational time in comparison to other models
4. A hybrid model of CNN and LSTM: This is a hybrid model designed recently and extensively used in stock price prediction. The model consists of both LSTM and CNN layers [31, 32].

These baseline models have their advantages and disadvantages. For example, FBProphet is the fastest model but underperforms in terms of error. Similarly, the hybrid model

[1] https://finance.yahoo.com

based on CNN and LSTM provides very little error but takes more time in predictions. A single neural network-based model performs well on one aspect of the problem while lags behind on the other front. Thus, we decided to exploit the good features of multiple networks. We have considered the following networks for their respective advantages in time series prediction.

3.1 | FastRNN-based hybrid model

FastRNN was proposed by Microsoft in 2018 [33]. The model is based on the Residual Network (ResNet) [34]. If we look at the structure of ResNet, then in this model, every network layer is not stacked over each other, but they form a residual mapping. It is shown that when compared with unreferenced mapping, the residual mapping is easier to optimise. The model is based on the visual geometric group model and modified by adding residual layers. ResNet is trained on 25.5 million parameters. This makes ResNet a very heavy network as well. FastRNN exploits the learned residual connections of ResNet and provides stable training along with comparable training accuracies at a much lower computational cost. The prediction accuracies of FastRNN are 19% higher than those of the regular RNN structures and a little less than those of the gated RNN.

Let $X = [x_1, \ldots, x_T]$ be the input vector where $x_t \epsilon R^D$ denotes the t-th step feature vector. So, for a multi-class RNN we aim to learn a function $F{:}R^{DxT}1, \ldots, L$ that is going to predict any of these L classes for a given point of data that is X. In a normal RNN architecture, hidden state vectors which are represented by

$$h_t = \tanh(Wx_t + Uh_{t-1} + b) \quad (1)$$

Learning U, and W in the architecture given above is arduous, as the gradient can have an exponentially large (in T) condition number. Unitary network-based ways can solve this problem by expanding the network size and in this process, they become significantly large and the model accuracy may not remain the same.

However, FastRNN utilises a simple weighted residual connection to stabilise the training by creating well-conditioned gradients. In particular, FastRNN updates the hidden state h_t as follows:

$$h'_t = \underline{\sigma}(Wx_t + Uh_{t-1} + b) \quad (2)$$

$$h_t = \alpha . h'_t + \beta . h_{t-1} \quad (3)$$

Here, α and β are trainable weights, which are parameterised by sigmoid function and limited in the range $0 < \alpha$, $\beta < 1$. σ is a non-linear function such as hyperbolic tangent, sigmoid, or Rectified Linear Unit (ReLU), and can vary across datasets.

FastRNN reconditions hidden state in a calculated manner with α, β limiting the extent to which the present attribute vector x_t updates the hidden state. Also, FastRNN has two more parameters than RNN and requires lesser computations, which is a very minute fraction of the per-stride computational complexity of the RNN. Unlike unitary methods [35, 36], FastRNN does not introduce any costly systemic constraints on U and hence scales well to huge datasets with typical optimisation methods. The proposed FastRNN-based model for stock market predictions is shown in Figure 1.

3.2 | FastRNN + CNN + Bi-LSTM-based hybrid model

The first model proposed in this work is designed to provide a faster computation time with reasonably good accuracy. The second aim of this work is to maintain fast computation with improved accuracy. In this direction, we propose a hybrid model that exploits the good features of multiple networks. We use FastRNN as our first network that will provide faster results and augment it with CNN and Bi-LSTM networks to improve the accuracy of prediction. Bi-LSTM was proposed in 1997 [37]. This model is designed to learn in both directions and is one of the best-suited models for sequence to sequence learning. For time series forecasting as well, the model has been proven to be a good fit [38–40]. The encouraging results from these recent works have inspired us to consider Bi-LSTM as our last network. We have taken the conventional CNN in between FastRNN and Bi-LSTM to stabilise the network. A CNN has similarities with an ANN but it takes presumptions about the input data, which allows it to attain greater invariance when encoding the properties of input data into the network. CNN needs to be trained on huge amounts of training data to generate deep learning models that attain higher generalisation accuracy. The proposed design of the hybrid model is shown in Figure 2.

In the proposed model, we have taken a three-stage pipeline consisting of FastRNN, CNN, and Bi-LSTM. In the first model, we have explained the added advantages of FastRNN for time series forecasting. The CNN layer is one-dimensional with an ReLU activation function. The kernel size is kept as 3 and keeping padding as 'same' followed by Maxpooling. 1-D CNN is used to extract the higher-level features only. These extracted features are fed to the final stage, that is, Bi-LSTM network. Bi-LSTM is followed by 2 dense layers. In the proposed model, the Bi-LSTM unit learns

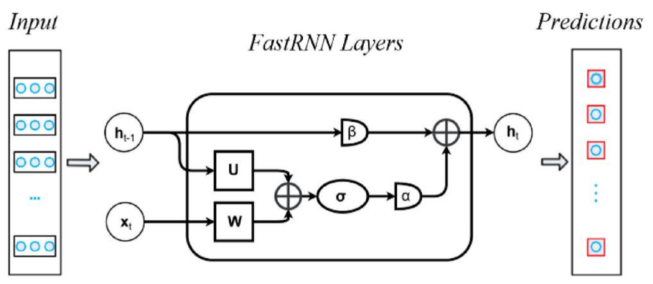

FIGURE 1 Fast Recurrent Neural Networks-based proposed model for faster stock values prediction

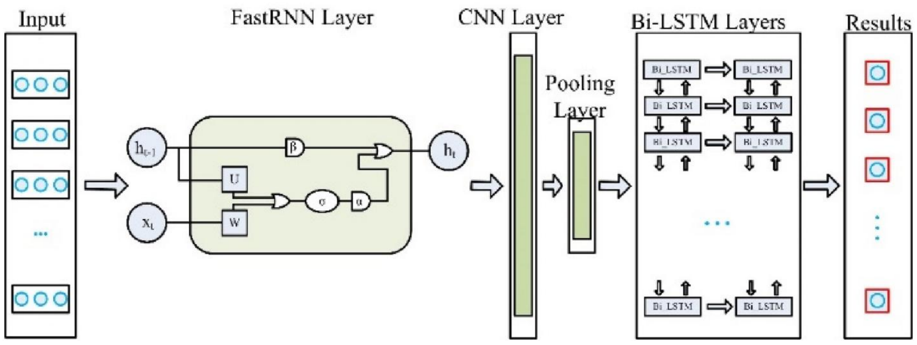

FIGURE 2 FastRNN, CNN, and bi-directional long short term memory based hybrid model for higher accuracy stock market predictions

from both the backward and forward sequences of data and uses concatenation to merge the two sequential outputs. There was no overfitting observed, that is why we have not added a dropout layer. Here, using Bi-LSTM also prevented us from the vanishing gradient problem. *Network parameters:* As mentioned before, we are using 430 stock values as input, and the 1-D CNN layer with the use of 3-kernel windows to return another sequence of smaller size. The output from the 1-D CNN is then fed into the Bi-LSTM layer group and is giving an output sequence of length 50. The output of this group is then fed to a dropout layer, which gives 25 units and 1 unit output, respectively. As it can be considered as a standard regression problem, we have used RMSE as the loss function to determine the error in the predicted versus actual data. We have to keep input and output rates identical and are critical for time-series forecasting.

Let us assume that [$x = x_1, x_2, x_3, \ldots, x_n$] is the one-dimensional input for the 1-D CNN layer. The equation makes a feature map after it gets convoluted with the convolution operator and is passed through a filter $W \epsilon Rf_d$, where f signifies inherent attributes from the input data throwing out as output. A new features set f_m from the new attributes f represented in the equation below:

$$hl_i^{fm} = \tanh\left(w^{fm} x_{i:i+f-1} + b\right) \tag{4}$$

Every set of features f uses the filter hl in the input defined by [$x_{1-f}, x_{2-f+1}, \ldots, x_{n-f+1}$]. This operation generates a feature map denoted by [$hl_1, hl_2, \ldots, hl_{n-f+1}$].

Convolution layer outputs are obtained as a sum of weighted inputs after multiple linear transformations. For a non-linear feature extraction problem, linear transformations do not perform with satisfactory success and thus we have to add non-linear activation functions. In this model, we have chosen the ReLU activation function which applies max (0, x) on each input. The output is down-sampled in the next step to reduce the information, so that the computation time can be improved. In our model, we have used max-pooling for that which is represented by hl = max (hl). Here, pooling helps the model to select the most relevant information and the output of the max-pooling layer can be denoted as follows:

$$x_i' = CNN(x_i) \tag{5}$$

where x_i is the data vector which is inputted to the CNN network and x_i' is the CNN network output which is then further passed to the Bi-LSTM network. To get the idea of Bi-LSTM, we add forget gate structure in the LSTM. The equation is denoted by

$$i_t = \sigma\left(W_i\left(\left[x_t, y_{t-1}\right]\right)\right) \tag{6}$$

$$f_t = \sigma\left(W_f\left(\left[x_t, y_{t-1}\right]\right)\right) \tag{7}$$

$$o_t = \sigma\left(W_o\left(\left[x_t, y_{t-1}\right)\right]\right)) \tag{8}$$

$$g_t = \tanh\left(W_g\left(\left[x_t, y_{t-1}\right]\right)\right) \tag{9}$$

$$c_t = f \odot c_{t-1} + i \odot g \tag{10}$$

$$y_t = o \odot \tanh(c_t) \tag{11}$$

where i is the input gate, f is the forget gate, o is the output gate, g, and c input modulation gate, respectively. Note that these are in n-dimensional real vectors. In Equations (6)–(8), the σ is a sigmoid function and W_i, W_f, W_o, and W_g are fully connected neural networks for the input, forget, output, and input modulation gates, respectively. The issue with the LSTM model is that it only considers information from one direction on a sequence which leads to an effective reduction of the LSTM model. Also, multi-directional information in the sequence can have valuable information data. Therefore, Bi-LSTM was developed and it combines backward and forward directions in the sequence.

Order for the forward LSTM is [$x_1, x_2, \ldots, x_n$]and for the backward LSTM is [$x_n, x_{n-1}, \ldots, x_1$]. Post-training, both the forward and the backward LSTMs separately are integrated by combining their outputs in the previous step, which is denoted in Equation (12) as,

$$yt = yF(t)yB(n-t+1) \tag{12}$$

where y_F and y_B are the outputs of the backward and forward LSTMs, respectively, while the notation denotes integration

TABLE 1 RMSE and computation time calculated for the state-of-the-art and the proposed models for Apple Inc. stock values

Model name	RMSE	Time (in s)
ARIMA [41]	0.796109	1.63
BiLSTM_Attention_CNN_BiLSTM [42]	0.234644	25.72292113
CNN_LSTM_Attention_LSTM [43]	0.214821	16.00164294
FBProphet [44]	0.935556	**0.659962893**
LSTM [45]	0.228731	13.28157353
LSTM_Attention_CNN_BiLSTM [42]	0.263613	19.96186757
LSTM_Attention_CNN_LSTM [45]	0.27994	17.32732081
LSTM_Attention_LSTM [45]	0.299334	19.28274226
LSTM_CNN_BiLSTM [46]	0.23489	16.76800251
FastRNN (proposed)	**0.202456**	**3.337492943**
FASTRNN_CNN_BiLSTM (proposed)	**0.205647**	**13.49208355**

Abbreviations: ARIMA, auto regressive integrated moving average; RMSE, root mean squared error.

operator such as a simple adder. The proposed model is developed on Google Colab. Details of experimental setup and results are provided in the next section.

4 | EXPERIMENTAL RESULTS AND DISCUSSIONS

4.1 | Experimental setup

We have used a free version of Google Colab with AMD EPYC 7B12 CPU as our execution environment with 12 GB (adjustable) assigned RAM, with one socket, two threads per core, 13,684K of L3 cache, CPU of around 2250 MHz, and with No Power level. We have developed our model using the Python programming language (Python 3.5). We used Keras (with Tensorflow backend) library for creating our deep learning models. To compile the data into a suitable format, we have used Pandas, and for dividing our data into test and train, we have used Numpy. For visualisation, we have the matplotlib library. The dataset and the corresponding codes are available at GitHub.[2]

4.2 | Results

We have used 430 stock values for training and 70 for testing. We have tested our models on the stock values of 4 companies, that is, Apple, Facebook, Nike, and Uber. These proposed models are compared with nine other state-of-the-art models. We have trained the models for 40 epochs. The RMSE values and the computation time for each of the nine models along with the two proposed models for four companies, that is, Apple, Facebook, Nike, and Uber are shown in Tables 1–4. These tables highlight the best values for each column.

The training process is fast and could be carried out on a CPU because the data-size was not very high. The values of loss functions for each of the 4 companies' data training are shown in the graphical form in Figures 3 and 4.

From the mentioned tables, Tables 1–4, it can be observed that the proposed models, not only have lesser RMSE but also they perform better in terms of their computation speed, which gives a clear indication that these models can be useful in making live next minute predictions of a stock price which will help the investor to buy stocks more wisely as the market can crash anytime due to any reason. The proposed models

[2]https://github.com/MilindYadav-97/Hybrid_FastRNN-for-stock-predictions

TABLE 2 RMSE and computation time calculated for the state-of-the-art models and the proposed models for Facebook Inc. stock values

Model name	RMSE	Time (in s)
ARIMA [41]	0.86664567	**1.60001**
BiLSTM_Attention_CNN_BiLSTM [42]	0.26834072	26.56901288
CNN_LSTM_Attention_LSTM [43]	0.15718991	16.61885238
FBProphet [44]	1.59050836	**0.405165672**
LSTM [45]	0.18714926	14.00746107
LSTM_Attention_CNN_BiLSTM [42]	0.22215487	20.65539312
LSTM_Attention_CNN_LSTM [45]	0.24017975	17.97418046
LSTM_Attention_LSTM [45]	0.29138532	19.86988878
LSTM_CNN_BiLSTM [46]	0.19898068	17.52065849
FastRNN (proposed)	**0.14932692**	**14.16171408**
FASTRNN_CNN_BiLSTM (proposed)	**0.15137204**	**3.447671652**

Abbreviations: ARIMA, auto regressive integrated moving average; RMSE, root mean squared error.

TABLE 3 RMSE and computation time calculated for the state-of-the-art models and the proposed models for Nike Inc. stock values

Model name	RMSE	Time (in s)
ARIMA [41]	0.465822234	**1.63001**
BiLSTM_Attention_CNN_BiLSTM [42]	0.106373725	26.56245756
CNN_LSTM_Attention_LSTM [43]	0.051930262	17.05163288
FBProphet [44]	0.550859082	**0.509964705**
LSTM [45]	0.052901	14.01486564
LSTM_Attention_CNN_BiLSTM [42]	0.062675555	20.4820962
LSTM_Attention_CNN_LSTM [45]	0.061914832	17.52905965
LSTM_Attention_LSTM [45]	0.102347907	19.73444676
LSTM_CNN_BiLSTM [46]	0.047966405	17.31208062
FastRNN (proposed)	**0.037727882**	**14.34483051**
FASTRNN_CNN_BiLSTM (proposed)	**0.039458341**	**3.807400703**

Abbreviations: ARIMA, auto regressive integrated moving average; RMSE, root mean squared error.

TABLE 4 RMSE and computation time calculated for the state-of-the-art models and the proposed models for Uber stock values

Model name	RMSE	Time (in s)
ARIMA [41]	0.163886332	**1.610950985**
BiLSTM_Attention_CNN_BiLSTM [42]	0.025170157	24.91984868
CNN_LSTM_Attention_LSTM [43]	0.023596089	15.85278392
FBProphet [44]	0.064013152	**0.489181757**
LSTM [45]	0.024660817	13.3820951
LSTM_Attention_CNN_BiLSTM [42]	0.031076263	19.3135848
LSTM_Attention_CNN_LSTM [45]	0.032733788	16.7529645
LSTM_Attention_LSTM [45]	0.027326071	18.67898536
LSTM_CNN_BiLSTM [46]	0.021864702	16.29291153
FastRNN (proposed)	**0.020682922**	**3.35441184**
FASTRNN_CNN_BiLSTM (proposed)	**0.021809913**	**18.18150258**

Abbreviations: ARIMA, auto regressive integrated moving average; RMSE, root mean squared error.

work best on multiple stocks, which can be seen in the mentioned tables that the proposed models have outperformed other classical and hybrid models in terms of both RMSE and computation time. There is a visualised comparative study in Figure 3, which shows how the validation loss (Mean Squared Error) on the validation dataset is improving after each epoch for all the studied stocks. Figure 4 shows how the model learning speed is increasing after each epoch, as we can see a horizontal line of training loss (Mean Squared Error) after five epochs. Here, Figure 5 shows the comparative study of the actual and predicted values of the baseline model and our proposed models for the next 20 min and it can be observed that the proposed models' predictions were closer than the actual stock values for each of the visualised stock.

5 | CONCLUSIONS

Financial data prediction can be very beneficial for companies and investors. Deep learning-based models have been quite effective for such predictions in recent years. In this work, we have proposed two models for this purpose. The first model that is based on FastRNN, can provide faster predictions when

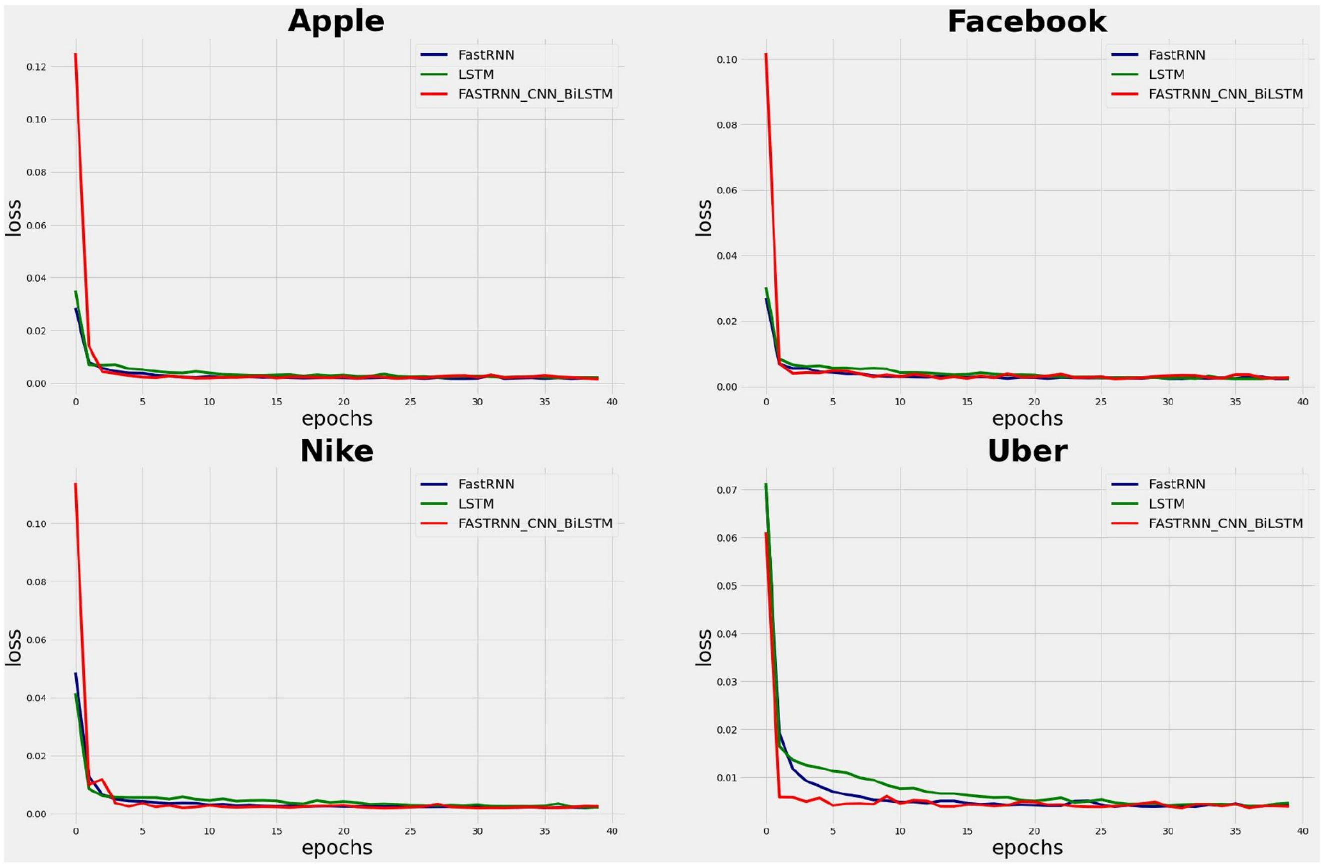

FIGURE 3 Comparison of plots of validation losses (mean squared error) of our proposed and baseline models at each epoch for all the studied stocks

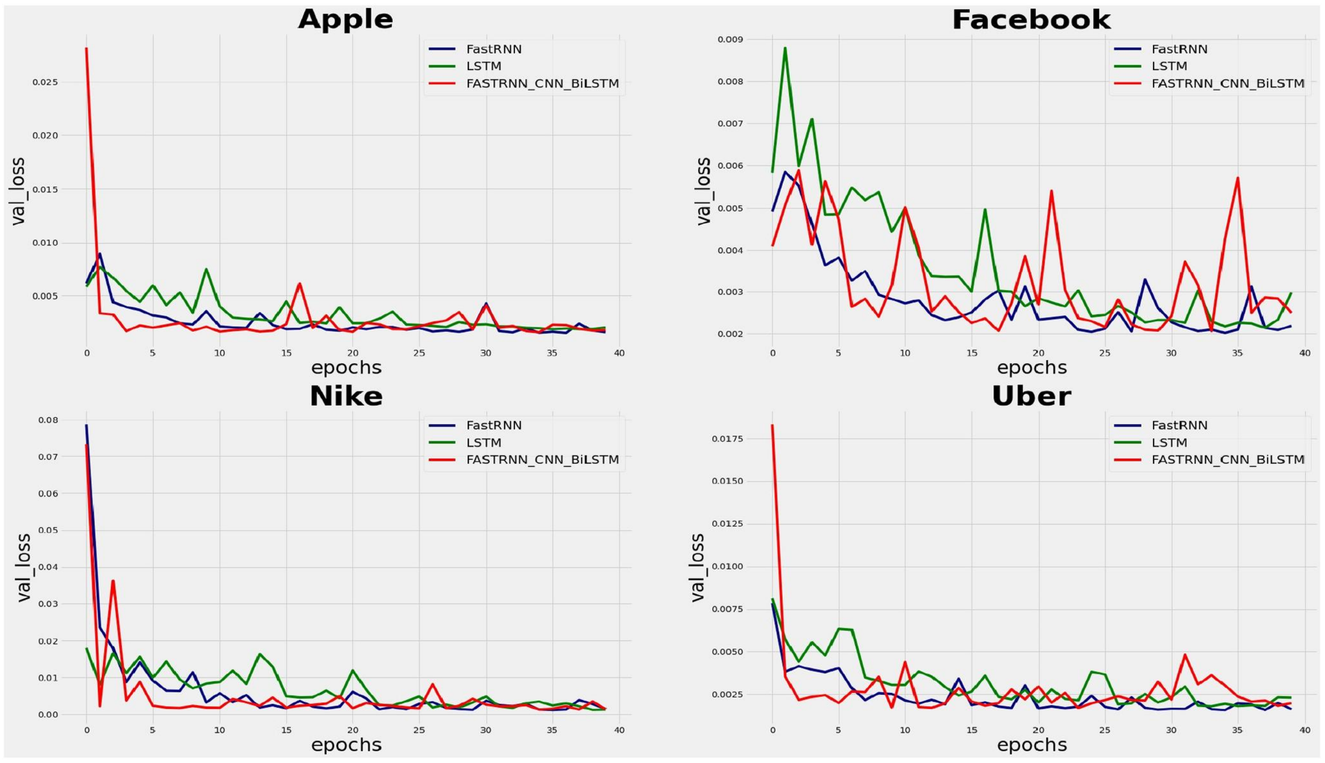

FIGURE 4 Comparative plots of training losses (mean square error) of our proposed and baseline models at each epoch for all the studied stocks

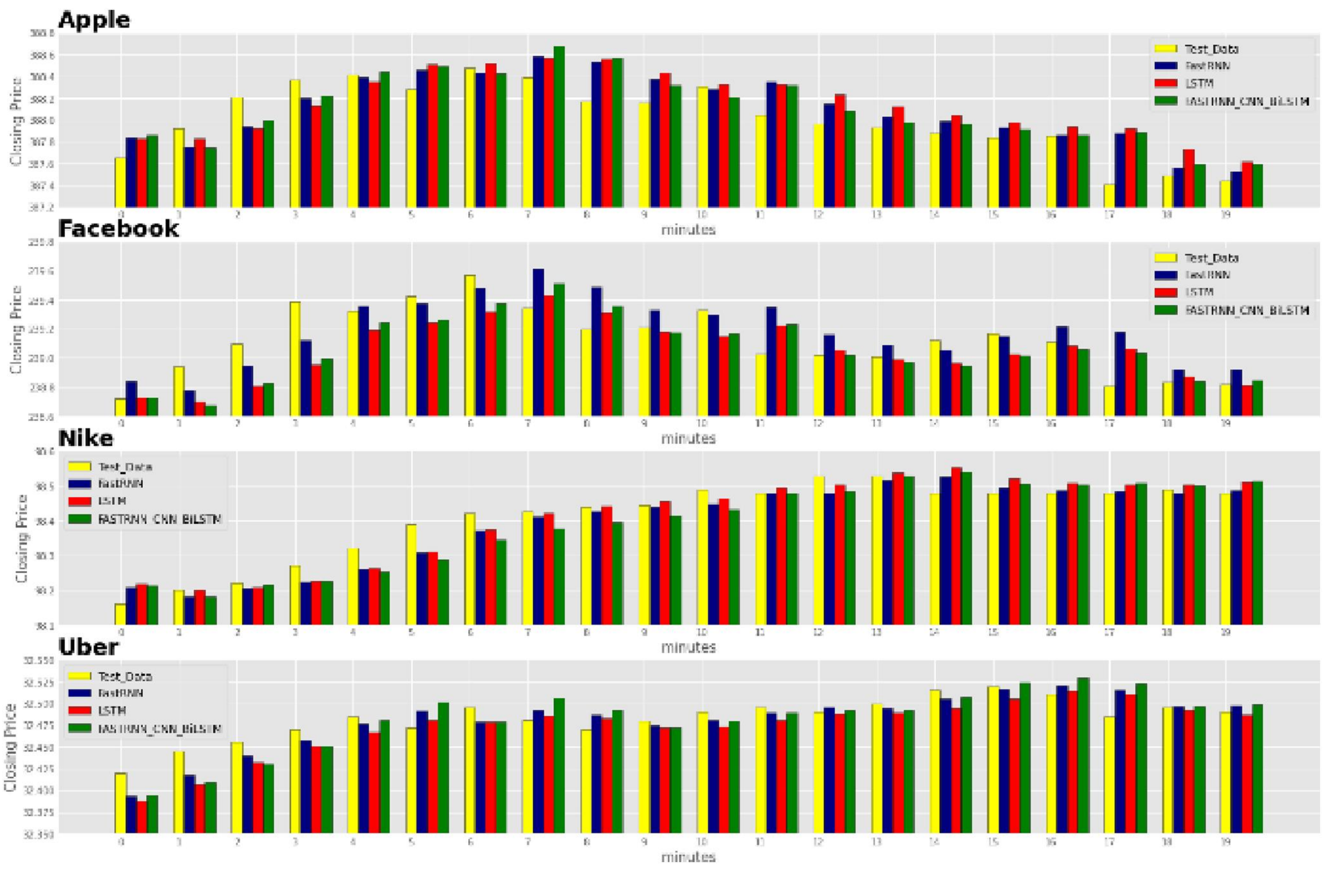

FIGURE 5 Predicted stock values with proposed model and baseline models

compared with other state-of-the-art models (except ARIMA and FBProphet). While improving the speed of prediction, it also provides better or at par RMSE values as well. Thus, also when compared with FBProphet and ARIMA, this is a better choice for a reliable model. The second model improves the first proposed model while keeping the speed almost the same. It compromises a bit on the speed of prediction but improves the RMSE values. These models can help people who have invested in stocks. Using these models, if suppose the investor gets to know that stock prices are crashing in the next minutes or so, then the investor can sell the stocks at the right time which can prevent the investor from heavy losses. Similarly, if the investor knows that in the next few minutes, stock prices will rise, then accordingly the investor can make a move. The only limitation with the proposed models is that we have not considered any other external factors on which stock prices might be dependent on, such as their demand and supply, geographical changes, company's profit/loss, etc. As we all know that stock prices are very volatile, it is better to take into account other factors as well such as region, geographical seasons, inflation, cost of raw products, market competition, demand, and supply, etc. on which stock prices are dependent. In the future, we will consider these other dependent attributes as well, which can affect the stock price values and will make our model more robust.

ACKNOWLEDGEMNTS

There is no funding source for this research. It is completed as part of the individual level with the support of the host institute, that is, the LNM Institute of Information Technology, Jaipur, India.

ORCID

Sandeep Saini https://orcid.org/0000-0002-8906-8639

REFERENCES

1. Hsu, M.-W., et al.: Bridging the divide in financial market forecasting: machine learners vs. financial economists. Expert Syst. Appl. 61, 215–234 (2016)
2. Ahmed, M.S., Cook, A.R.: Analysis of Freeway Traffic Time-Series Data by Using Box-Jenkins Techniques, Vol. 722. Transportation Research Board, Washington, D.C. (1979)
3. Huang, S.-J., Shih, K.-R.: Short-term load forecasting via ARMA model identification including non-Gaussian process considerations. IEEE Trans. Power Syst. 18(2), 673–679 (2003)
4. Sharma, A., Bhuriya, D., Singh, U.: Survey of stock market prediction using machine learning approach. In: 2017 International Conference of Electronics, Communication and Aerospace Technology (ICECA), vol. 2, pp. 506–509. IEEE, New York, 20–22 April 2017. https://ieeexplore.ieee.org/document/8212715
5. Franses, P.H., Van Dijk, D.: Forecasting stock market volatility using (non-linear) GARCH models. J. Forecast. 15(3), 229–235 (1996)
6. Tay, F.E.H., Cao, L.: Application of support vector machines in financial time series forecasting. Omega. 29(4), 309–317 (2001)
7. Egeli, B., Ozturan, M., Badur, B.: Stock market prediction using artificial neural networks. Decis. Support Syst. 22, 171–185 (2003)
8. Enke, D., Thawornwong, S.: The use of data mining and neural networks for forecasting stock market returns. Expert Syst. Appl. 29(4), 927–940 (2005)
9. Abraham, A., Nath, B., Mahanti, P.K.: Hybrid intelligent systems for stock market analysis. In: International Conference on Computational Science, pp. 337–345. Springer, Heidelberg, 28–30 May 2001. https://link.springer.com/content/pdf/10.1007/3-540-45718-6_38.pdf
10. Armano, G., Marchesi, M., Murru, A.: A hybrid genetic-neural architecture for stock indexes forecasting. Inf. Sci. 170(1), 3–33 (2005)
11. Fu, J., et al.: Stock prediction using FCMAC-BYY. In: International Symposium on Neural Networks, pp. 346–351. Springer, Heidelberg (2007)
12. Choudhry, R., Garg, K.: A hybrid machine learning system for stock market forecasting. World Acad. Sci. Eng. Technol. 39(3), 315–318 (2008)
13. Tsai, C.F., Wang, S.P.: Stock price forecasting by hybrid machine learning techniques. In: Proceedings of the International MultiConference of Engineers and Computer Scientists, vol. 1, pp. 60. IAENG, Hong Kong, 14-16 March 2018. http://www.iaeng.org/publication/IMECS2018/
14. Ding, X., et al.: Deep learning for event-driven stock prediction. In: Twenty-Fourth International Joint Conference on Artificial Intelligence, Buenos Aires, 25–31 July 2015
15. Akita, R., et al.: Deep learning for stock prediction using numerical and textual information. In: 2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS), Okayama, 26–29 June 2016, pp. 1–6. IEEE, Piscataway, New Jersey (2016)
16. Fischer, T., Krauss, C.: Deep learning with long short-term memory networks for financial market predictions. Eur. J. Oper. Res. 270(2), 654–669 (2018)
17. Hoseinzade, E., Haratizadeh, S.: CNNpred: CNN-based stock market prediction using a diverse set of variables. Expert Syst. Appl. 129, 273–285 (2019)
18. Gudelek, M.U., Boluk, S.A., Ozbayoglu, A.M.: A deep learning based stock trading model with 2-D CNN trend detection. In: 2017 IEEE Symposium Series on computational Intelligence (SSCI), 27 November–1 December 2017, Honolulu, pp. 1–8. IEEE, Piscataway, New Jersey (2017)
19. Eapen, J., Bein, D., Verma, A.: Novel deep learning model with CNN and bi-directional LSTM for improved stock market index prediction. In: 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), pp. 264–270. IEEE, New York, 6–8 January 2020
20. Selvin, S., et al.: Stock price prediction using LSTM, RNN and CNN-sliding window model. In: 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 13–16 September 2017, Manipal, pp. 1643–1647. IEEE, Piscataway, New Jersey (2017)
21. Nabipour, M., et al.: Predicting stock market trends using machine learning and deep learning algorithms via continuous and binary data: a comparative analysis. IEEE Access. 8, 150199–150212 (2020)
22. Nabipour, M., et al.: Deep learning for stock market prediction. Entropy. 22(8), 840 (2020)
23. Ecer, F., et al.: Training multilayer perceptron with genetic algorithms and particle swarm optimization for modeling stock price index prediction. Entropy. 22(11), 1239 (2020)
24. Nosratabadi, S., et al.: Data science in economics: comprehensive review of advanced machine learning and deep learning methods. Mathematics. 8(10), 1799 (2020)
25. Shamshirband, S., Rabczuk, T., Chau, K.-W.: A survey of deep learning techniques: application in wind and solar energy resources. IEEE Access. 7, 164650–164666 (2019)
26. Shamshirband, S., et al.: A review on deep learning approaches in healthcare systems: taxonomies, challenges, and open issues. J. Biomed. Inform, 113, 103627 (2020)
27. Gers, F.A., Eck, D., Schmidhuber, J.: Applying LSTM to time series predictable through time-window approaches. In: Neural Nets WIRN Vietri-01, pp. 193–200. Springer, Heidelberg, 21–25 August 2001
28. Yadav, K., et al.: Bi-LSTM and ensemble based bilingual sentiment analysis for a code-mixed Hindi-English social media text. In: 2020 IEEE 17th India Council International Conference (INDICON), 11–13 December 2020, Delhi, pp. 1–6. IEEE, Piscataway, New Jersey (2020)

29. Saini, S., Sahula, V.: Neural machine translation for English to Hindi. In: 2018 Fourth International Conference on Information Retrieval and Knowledge Management (CAMP), Kota kinabalu, Malaysia, pp. 1–6. IEEE, New York, 26–28 March 2018. https://ieeexplore.ieee.org/document/8464781
30. Taylor, S.J., Letham, B.: Forecasting at scale. Am. Stat. 72(1), 37–45 (2018)
31. Kim, T., Kim, H.Y.: Forecasting stock prices with a feature fusion LSTM-CNN model using different representations of the same data. PLoS One. 14(2), e0212320 (2019)
32. Yadav, K., et al.: Bilingual sentiment analysis for a code-mixed Punjabi English social media text. In: 2020 5th International Conference on Computing, Communication and Security (ICCCS), 14–16 October 2020, Patna, pp. 1–5. IEEE, Piscataway, New Jersey (2020)
33. Kusupati, A., et al.: FastGRNN: a fast, accurate, stable and tiny kilobyte sized gated recurrent neural network. In: NIPS'18: Proceedings of the 32nd International Conference on Neural Information Processing Systems, San Diego, pp. 9017–9028 Montréal, 8 December 2018. https://proceedings.neurips.cc/paper/2018/hash/ab013ca67cf2d50796b0c11d1b8bc95d-Abstract.html
34. He, K., et al.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 27–30 June 2016, Las Vegas, pp. 770–778. IEEE, Piscataway, New Jersey (2016)
35. Arjovsky, M., Shah, A., Bengio, Y.: Unitary evolution recurrent neural networks. In: International Conference on Machine Learning, pp. 1120–1128 (2016)
36. Zhang, J., Lei, Q., Dhillon, I.S.: Stabilizing gradients for deep neural networks via efficient SVD parameterization. arXiv Prepr. arXiv1803.09327 (2018)
37. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE Trans. Signal Process. 45(11), 2673–2681 (1997)
38. Siami-Namini, S., Tavakoli, N., Namin, A.S.: The performance of LSTM and BiLSTM in forecasting time series. In: 2019 IEEE International Conference on Big Data (Big Data), pp. 3285–3292 (2019)
39. Kim, J., Moon, N.: BiLSTM model based on multivariate time series data in multiple field for forecasting trading area. J. Ambient Intell. Humaniz. Comput. 10(4), 1–10 (2019)
40. Long, J., et al.: An integrated framework of deep learning and knowledge graph for prediction of stock price trend: an application in Chinese stock exchange market. Appl Soft Comput. 91, 106205 (2020)
41. Ariyo, A.A., Adewumi, A.O., Ayo, C.K.: Stock price prediction using the ARIMA model. In: 2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation, pp. 106–112. IEEE, New York, 26–28 March 2014. https://ieeexplore.ieee.org/document/7046047
42. Wang, M., Cheng, J., Zhai, H.: Life prediction for machinery components based on CNN-BiLSTM network and attention model. In: 2020 IEEE 5th Information Technology and Mechatronics Engineering conference (ITOEC), 12–14 June 2020, Chongqing, pp. 851–855. IEEE, Piscataway, New Jersey (2020)
43. Zhang, Y., et al.: A text sentiment classification modeling method based on coordinated CNN-LSTM-attention model. Chinese J. Electron. 28(1), 120–126 (2019)
44. Chikkakrishna, N.K., et al.: Short-term traffic prediction using SARIMA and FbPROPHET. In: 2019 IEEE 16th India Council International conference (INDICON), 13–15 December 2019, Rajkot, pp. 1–4. IEEE, Piscataway, New Jersey (2019)
45. Qiu, J., Wang, B., Zhou, C.: Forecasting stock prices with long-short term memory neural network based on attention mechanism. PLoS One. 15(1), e0227222 (2020)
46. Le, T., et al.: Improving electric energy consumption prediction using CNN and Bi-LSTM. Appl. Sci. 9(20), 4237 (2019)

How to cite this article: Yadav, K., Yadav, M., Saini, S.: Stock values predictions using deep learning based hybrid models. CAAI Trans. Intell. Technol. 7(1), 107–116 (2022). https://doi.org/10.1049/cit2.12052