Stock Price Prediction on Daily Stock Data using Deep Neural Networks

Sneh Jain[1], Dr. Roopam Gupta[2], Dr. Asmita A. Moghe[3]

Abstract— The price of a stock is volatile and complex in nature which makes its prediction a difficult task. This paper plans to predict the prices of Tata Consultancy Services (TCS) and Madras Rubber Factory Limited (MRF) stocks on a short-term basis. In this paper, a comparative analysis of various Deep Neural Network techniques applied for a stock price prediction application is done. The networks used are pertinent to the problem include Convolutional Neural Networks, Long Short-Term Memory Networks and Conv1D-LSTM. The different neural network models are trained on daily stock price data which includes Open, High, Low, and Close price values. These are used to predict the next day closing price. From the last 5 days of data, the prediction is made. Results of different models are compared with each other. In this paper, a deep neural network Conv1D-LSTM is proposed which is based on the combining of layers of two different techniques – CNN and LSTM to predict the price of a stock. The performance of the models is evaluated using RMSE, MAE and MAPE. These errors in Conv1D-LSTM model are found to be very low compared to CNN & LSTM. For stock price prediction, Conv1D-LSTM network is found to be effective, depending on the nature of stock hyper-parameters may require some variations.

Keywords— Time-series, Stock Price Prediction, Deep Learning, Deep Neural Networks, LSTM, CNN, Sliding window, 1D Convolutional - LSTM network**.**

I. Introduction

Researchers in recent years have been using deep neural networks broadly for the application of regression, classification and prediction. Deep neural networks have been making progress so well because of the availability of data and the scale at which computation is available [1]. A sequence of numerical data points taken at equally spaced points in time in serial order is known as time-series data. One of the applications of deep learning includes time-series prediction, which is predicting the future values of an object at a certain time. Prediction can be mainly classified as short-term (prediction for seconds, minutes, and days) and long-term (prediction for more than one year or beyond).

This paper deals with one particular time-series prediction which is related to the Financial sector called Stock Price Prediction. In this Time-Series the variable is the stock price. The economic benefits can be achieved easily by predicting the development of financial mechanisms like Stocks. The behavior of stock is highly volatile and complex in nature. The frequent fluctuation in stocks makes it difficult to predict its future movement. Stock market requires prior knowledge so that the investment decision can be made wisely. The techniques involved in the time-series analysis of data related to finance and stocks have gained importance due to their nature of helping in maximizing the profits while keeping the low potential of risk.

The recent development in the analysis of time-series involved the incorporation of deep neural networks [2, 3] such as CNN, RNN, LSTM networks. An Ensemble of deep neural networks are also applied to the problem of forecasting in stocks [4] but they involve the training of each expert separately and then to obtain the results averaging methods are applied. Instead of training different neural networks separately, one can combine the layers of different models in a single deep neural network. In this paper, the methodology proposed is based on combining the layers of different techniques into a single deep neural network while using less number of features for training.

II. Related Work

The existing approaches are mainly divided into two categories: Fundamental Analysis and Technical Analysis, to predict the future movement of a stock price. Interest rates, various ratios of prices (such as Price to earnings ratio), parameters related to economics form the basis for the Fundamental Analysis whereas past price values, historical data, volumes traded over a particular time, various moving averages form the basis of Technical Analysis[5].

Then comes the machine learning approaches which includes various machine learning techniques like Regression, ARIMA model, SVM [6], random-forest [7] as well as Neural Networks approaches. Usually, classification techniques like the k-nearest neighbor, Naïve Bayes algorithm are used to predict the stock trend [8] rise or fall but this paper tries to

[1]Department of Information Technology,
University Dual Degree Integrated Post Graduate Programme, RGPV, Bhopal, Madhya Pradesh, India
snehjain111@gmail.com

[2]Department of Information Technology,
University Institute of Technology, RGPV, Bhopal,
Madhya Pradesh, India
roopamgupta@rgtu.net

[3]Department of Information Technology,
University Institute of Technology, RGPV, Bhopal,
Madhya Pradesh, India
aamoghe@rgtu.net

solve the problem as a regression by making use of Deep Neural Networks. In many cases, SVM has performed much better than other models and even some neural network also. The neural network approaches had also been investigated for stock markets of India in [9] where for performance analysis metrics like RMSE, MAE, MAPE had been used. These machine learning and deep learning models require data from which they learn since they are based on supervised learning approaches. The basic steps commonly involved in the above-mentioned techniques are as follows: data collection and preprocessing steps, and then the training of the model is done. After the training, prediction and evaluation of the model is carried out.

A. Techniques used in Modern Approach

This section briefly explains the networks used in the recent approaches. These techniques are based on the deep neural networks.

a) Convolutional Neural Networks

These networks are used when data related to the task is spatial in nature or exists in a grid-like topology. The basic components of a Convolutional Layers are:

- Convolutional Layer: A filter of size k is run across the input (image) to get a convolved output.
- Pooling Layers: each feature map or the convolved output is then subsampled sometimes with a mean or max pooling layer, this helps in extracting the important features from the data input.
- Fully connected layer with output layer: the output of the above layers is then sent to this layer so that the problem can be solved in a manner as done with traditional Neural Network.

The use of Convolutional layers helps us prevent the various preprocessing steps as the features which are important can be extracted during the process of learning.

b) LSTM (Long Short-Term Memory) Networks

LSTM introduced in 1997 in [2], is an enhanced version of Recurrent Neural Network (RNN). RNN addresses the limitations of FFNN. RNNs allow information to persist in the network by making use of feedback loop. The drawback of the RNN was its vanishing gradient problem which was conquered in LSTM Network. These networks are used when the data is of sequential form. When time information of the data is more important than the spatial content of each individual frame in a dataset then this network is used. The limitation of the Feed-Forward Neural Networks is the consideration of current inputs and notion of order of time is absent, this is not the case with Recurrent Neural Networks or the LSTM networks. In this network, current inputs and previously learned inputs are taken into considerations. LSTM network consists of units called as Memory cell unit or memory cell in the place of hidden layers. These cells have three gates known as input gate, forget gate, output gate. These gates in LSTM cell carefully regulates the memory cell ability to add or remove the information from cell state.

In recent studies, it has been found that the deep neural networks (like CNN, RNN, and LSTM) are better in handling non-linear models [10]. Recent works imply the use of modern approaches like deep learning models in stock prediction by researchers which are not yet fully exploited.

III. Methodology

A. Proposed Methodology

To design a NN model for the prediction of stock price on input data set some basic steps like Data Collection and preprocessing, Model building and training, evaluation of prediction from those NN models are performed. Each NN model based on CNN and LSTM layers undergo these steps. Each of these steps are explained in this sections.

For experiment purpose, closing prices of daily trading data are predicted. Flow chart of the steps involved in proposed method for each NN model is shown in Fig. 1.

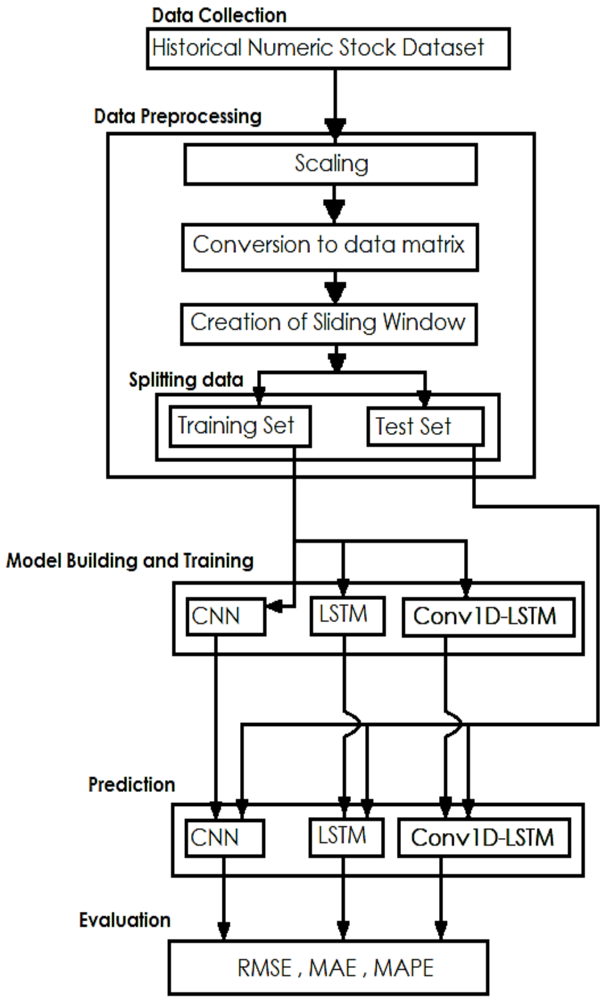

Fig. 1 Flow chart for CNN, LSTM and Conv1D-LSTM models.

A financial time series is a collection of prices such as stock, currency and commodity, obtained sequentially in time. The historical numeric data of stock price is collected from the data source. The variables in the data are stock prices. More than five years of daily prices have been collected and will be used in the following experiment.

After the collection of the dataset, a linear transformation is used on the attributes Open, High, Low, and Close to scale down the values between 0 and 1. Then, the data is converted into matrix form for faster computation and sliding-window approach is applied to the matrix.

By making use of sliding-window approach a sliding window of 6 days (5 days overlapped) is used, which consist of Open, High, Low, and Close prices of those days. From a sliding window, 5 days of data is used as input (x_train) and the output (y_train) is the closing price of the sixth day. This is how the data is developed for the neural network. The time-series dataset is split into 80% for training and 20% for testing. Further, the training set is divided into training and validation sets which include 20% data for validation from the training set during training.

After the preprocessing step, the sequential deep neural networks models are developed and are trained on the training data. The models are trained on daily stock price data, to make short-term predictions for a day in advance. The architectures of each DNN models used are represented in Sec. III (B).

When the training is completed the models are used to obtain the predictions on the test set. The models are then evaluated on the basis of the outcomes from prediction. The networks are evaluated using RMSE as loss function. The metrics used for evaluation of prediction are MAE and MAPE which are used after performing the inverse linear transformation to actual (y_{test}) and predicted values of output. The Mean Absolute Error (MAE) represented as EMAE is calculated using Eq.1.

$$E_{MAE} = \frac{\sum abs\left[Y_{actual} - Y_{pred}\right]}{n} \qquad (1)$$

Similarly, Mean Absolute Percentage Error (MAPE) represented as E_{MAPE} is calculated using Eq. 2.

$$E_{MAPE} = \frac{\sum \frac{abs\left[\,Y_{actual} - Y_{pred}\,\right]}{Y_{actual}}}{n} \times 100 \qquad (2)$$

Where Y_{actual} is the actual value, Y_{pred} is the predicted value and n is the no. of samples in the test dataset.

B. Network Architectures

This section represents the architectures of the neural networks used in the experiment. As per the proposed method, following neural networks have been used to the build deep neural network models are:

- Convolutional Neural Network (CNN)
- Long Short-Term Memory Network (LSTM)
- Conv1D-LSTM network: (combination of CNN and LSTM layers with base layers of CNN.)

a) CNN Architecture

Fig. 2 describes the layers present in between the input and output layers of CNN network. This neural network involves dense, Convolutional and MaxPooling layers.

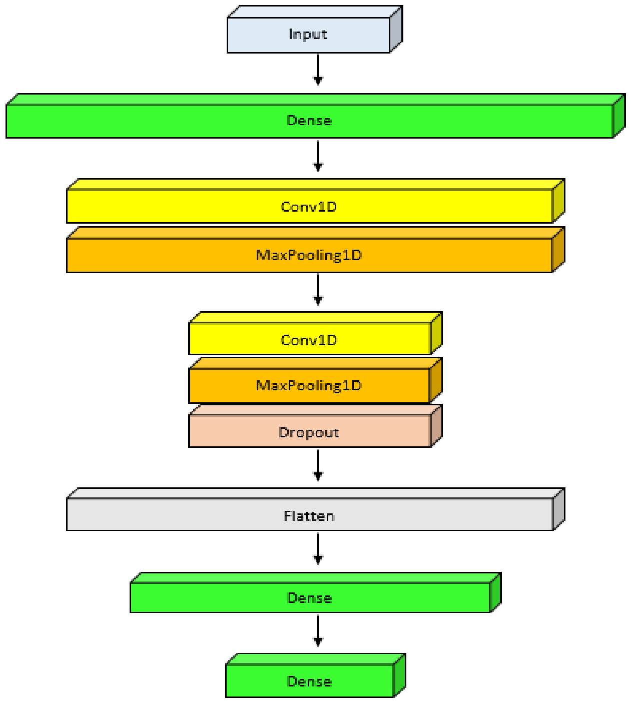

Fig. 2 CNN Architecture

b) LSTM Architecture

Fig. 3 describes the layers present in between the input and output layers of LSTM network. This neural network involves dense and LSTM layers.

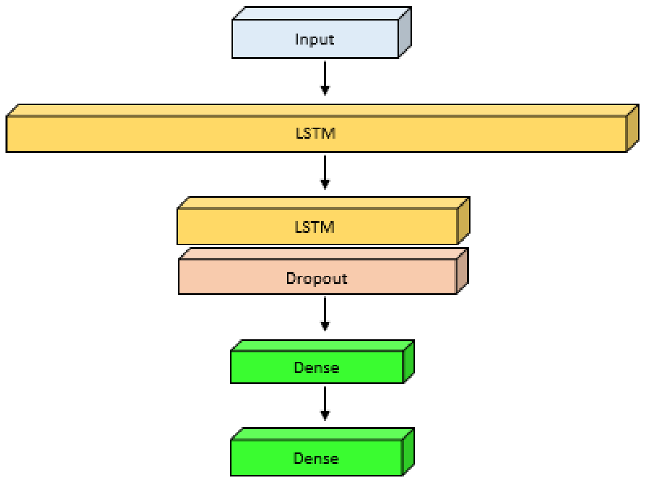

Fig. 3 LSTM Architecture

c) Conv1D-LSTM Architecture

The proposed methodology is a combination model based on CNN and LSTM network techniques. This scheme is usually used to exploit the advantages of techniques which are combined. In this model, 1D convolutional layers are used as preprocessing step to extract the important features, which are then passed on to LSTM layers for the process.[11].

Fig. 4 describes the layers present in between the input and output layers of the Conv1D-LSTM network. This neural network involves dense, Convolutional, MaxPooling and LSTM layers.

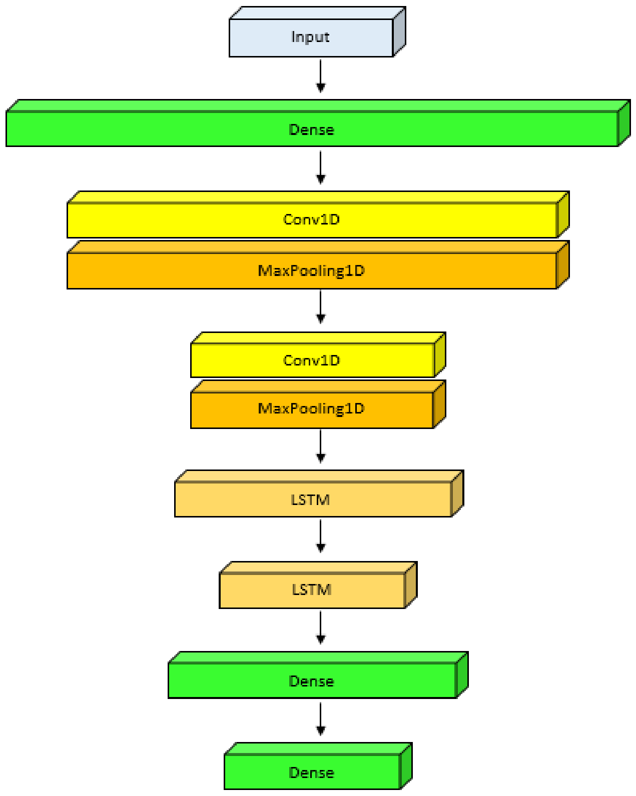

Fig. 4 Conv1D-LSTM Architecture

Fig. 2 – 4 represent the architectures of different models which includes various layers of sequential networks from input to output: Dense, Conv1D, MaxPool, Flatten, Dropout, and LSTM.

IV. Implementation & Experimental Result

This section represents the implementation details, results and observation obtained in the experiment. The experiment was performed on Tata Consultancy Services (TCS) and Madras Rubber Factory Limited (MRF) which are listed in National Stock Exchange of India (NSE) [11].

For this experiment, a high-level neural network API known as Keras is used with Tensorflow. Tensorflow is an open source machine learning framework and also the backend on which Keras is implemented. The API and framework are both written in python. The program for this experiment is implemented in Python-3.6.4, Tensorflow-1.8.0 and Keras-2.1.6. . The CPU platform used for training is Intel Core i5 with RAM 4.00GB.

The historical data of TCS and MRF are collected from the API of the financial source known as Quandl [12]. Quandl is a platform that serves financial, economic, and alternative data for various investment purposes. The dataset is from 01/01/2013 to 18/05/2018 and it’s consists of entries like Dates, the Opening price of the day, the High and Low price of the day, Close price and Adj. Close value of the day, Volume of stock traded, and Turnover (in lacs). In this experiment, the values of Open, High, Low, and Close are used for the purpose of prediction and sliding-window approach is used for training the neural networks [5]. For the sake of comparison, the number of epochs used for training is same in all NN models.

A. Learning Model Details

This section will give details about hyper-parameters of different models used in the experiment. The models used in this experiment are CNN, LSTM and Conv1D-LSTM as given in sub-sections of section III (B) Network Architectures.

a) Model Details for TCS

Hyper-parameters chosen for layers of CNN Network (which are taken from top to bottom) as are as follows:

- Units of Dense = [128, 100, 1] in respective Dense Layers
- No. of filters = [112, 64] in respective Conv1D Layers
- Pool size = [2, 1] in respective MaxPooling Layers
- Kernel size = 1
- Activation Function: ReLU

Hyper-parameters chosen for layers of LSTM network (which are taken from top to bottom) are as follows:

- No. of cells in LSTM layers = [128, 32] respectively
- Units of Dense = [16, 1] in respective Dense Layers
- Activation Function: ReLU

Hyper-parameters chosen for layers of Conv1D-LSTM Network (which are taken from top to bottom) are as follows:

For Base Convolutional Layers
- Units of Dense = [128, 32, 1] in respective Dense Layers
- No. of filters = [80, 48] in respective Conv1D Layers
- Pool size = 2 in respective MaxPooling Layers
- Kernel size = 1
- Activation Function: ReLU

For LSTM layers
- No. of cells in LSTM layers = [32, 16] respectively
- Activation Function: ReLU

b) Model Details for MRF

The hyper-parameters of CNN and LSTM models are kept same for both TCS and MRF. Hyper-parameters of the combination model i.e. Conv1D-LSTM used for MRF differs from hyper-parameters used for TCS stock.

Hyper-parameters chosen for layers of Conv1D-LSTM Network (which are taken from top to bottom) are as follows:

For Base Convolutional Layers

- Units of Dense = [128, 32, 1] in respective Dense Layers
- No. of filters = [24, 48] in respective Conv1D Layers
- Pool size = 2
- Kernel size = 1
- Activation Function: ReLU

For LSTM layers

- No. of cells in LSTM layers = [40, 32] respectively
- Activation Function: ReLU

Other Hyper-parameters which are kept same in the fit function of all the DNN models are as follows:

- Batch size = 128
- Epochs = 35 (for TCS), 25 (for MRF)
- Validation split = 0.2
- Optimizer = Adam
- Dropout rate = 0.2
- Loss function = mean_square_error
- Metrics = mean_absolute_error

The other hyper-parameters indicate that all the models will start iterations on the training data in batches of 128 samples, each iteration of DNN model over all the training data is called an epoch which is taken equal to 35 and 25 for TCS & MRF respectively. The gradients of the weights with regard to the loss on a batch will be computed after each iteration and weights will be updated accordingly in the DNN. Validation split represents the percentage of data taken for validation from training data.

To update the network weights during training, an optimization algorithm known as Adam [13] is used in which for different parameters an adaptive learning rates are computed. To prevent the over-fitting in neural networks a regularization techniques known as dropout is used, dropout rate represents the percentage of nodes dropped for each iteration and sample during training.

B. Results on TCS Dataset

Fig. 5, 6, 7 represents the plots of epochs vs. loss for CNN, LSTM & Conv1D-LSTM network represented in Sec. III (B). Here it is seen that after some epochs the validation loss becomes less than the training loss when fitted on TCS dataset. The models are modifying over time, and loss over the last batches is much lower than the initial batches during the training. The mean of the losses over each batch of training is described as training loss. The validation loss is found to be less than training loss because it is calculated at the end of each epoch and before the start of a new epoch when the model is not optimizing.

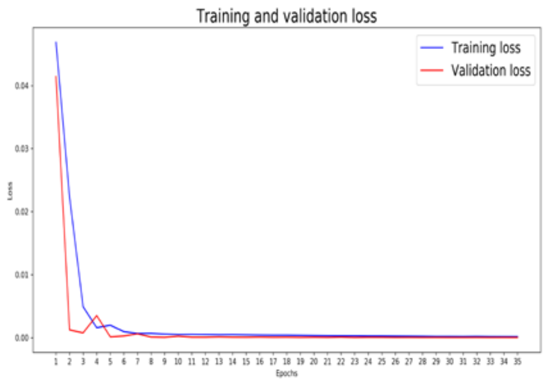

Fig. 5 Training and Validation Loss of CNN Model (TCS Data).

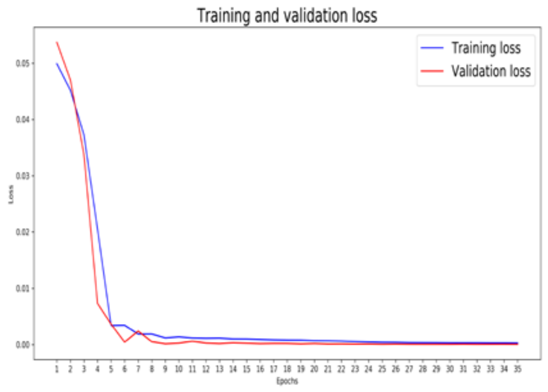

Fig. 6 Training and Validation Loss of LSTM Model (TCS Data).

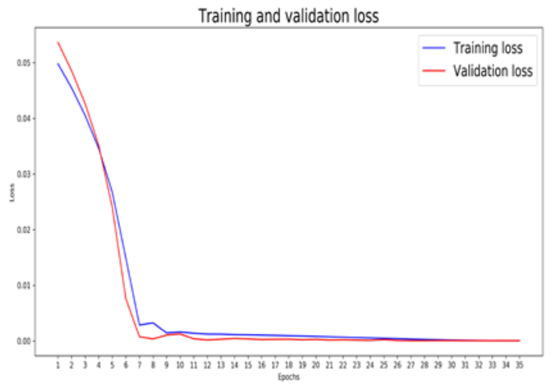

Fig. 7 Training and Validation Loss of Conv1D-LSTM Model (TCS Data).

The plots obtained when close values are predicted on the Training set of TCS containing 1060 samples are shown in

Fig. 8 – 10 for CNN, LSTM & Conv1D-LSTM Network respectively. It is evident that the LSTM model in fig. 9 and Conv1D_LSTM model in fig. 10 are able to capture the dynamics of time-data better than the CNN model in fig. 8.

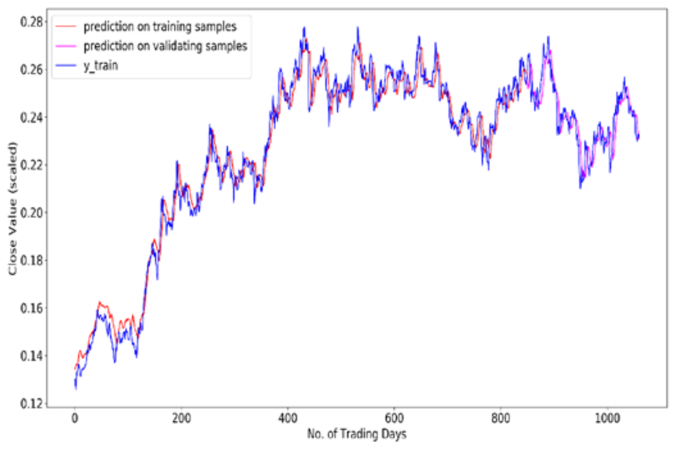

Fig. 8 Close values of TCS in CNN model after 35 epochs

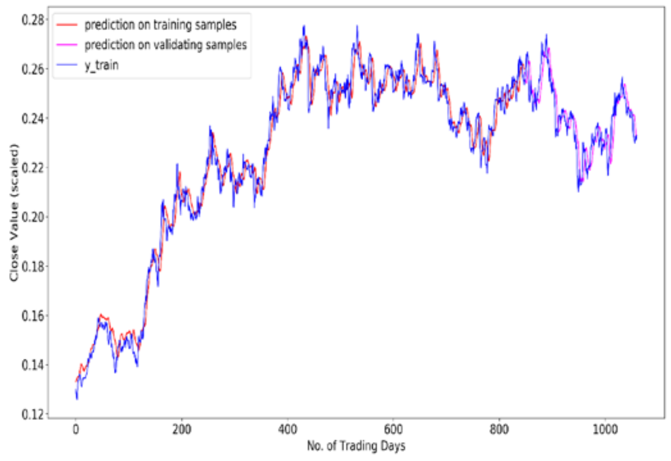

Fig. 9 Close values of TCS in LSTM model after 35 epochs

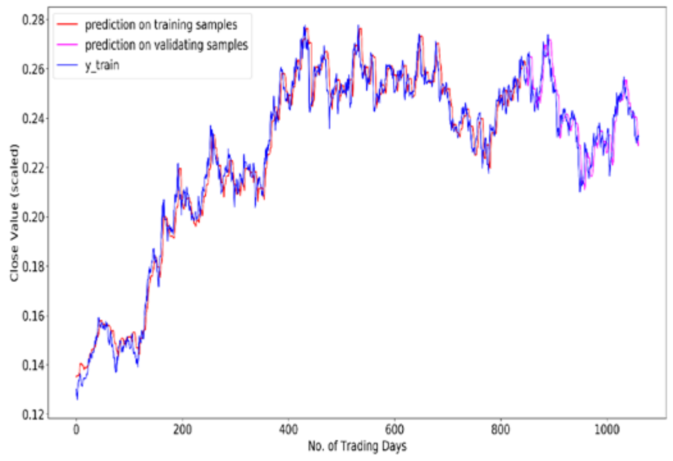

Fig. 10 Close values of TCS in Conv1D-LSTM model after 35 epochs

The plots obtained when close values are predicted on the Testing set of TCS containing 265 samples are shown in Fig. 11 – 13 for CNN, LSTM & Conv1D-LSTM network. The Predicted values are seen to closely follow the actual values which are clearly visible in the plots. Thus, the models are adept enough to follow the direction or the trend of the stock.

Since this problem is treated as a regression problem, the models cannot predict the actual values of interest. It can be observed from the Fig. 11 – 13 that CNN and Conv1D-LSTM are more capable of capturing the spatiotemporal content of the stock then LSTM, as the LSTM plot is much smoother than that of a plot of other two(CNN and Conv1D-LSTM) as shown in Fig. 12.

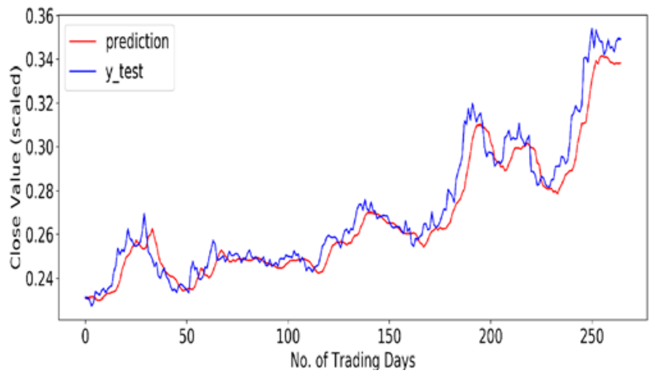

Fig. 11 Prediction of Close values (TCS) in CNN model on Testing Set

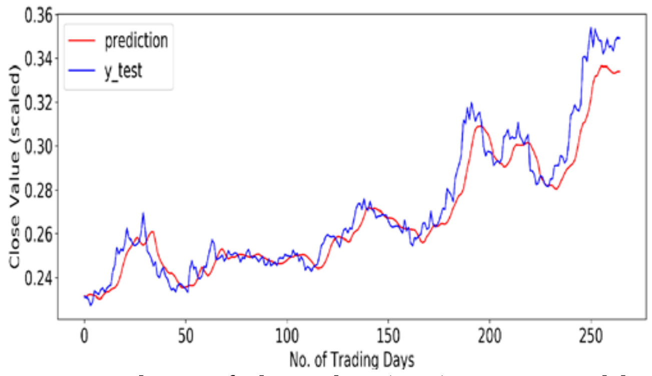

Fig. 12 Prediction of Close values (TCS) in LSTM model on Testing Set

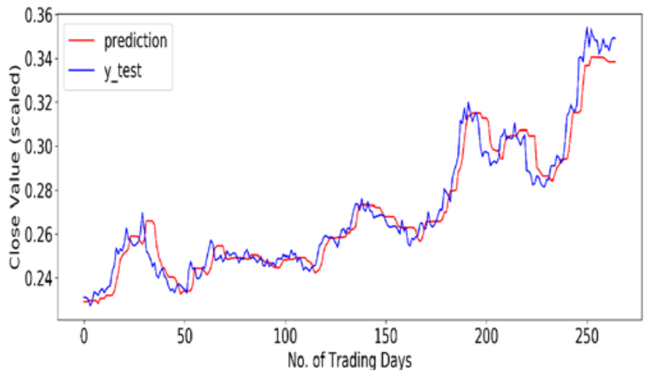

Fig. 13 Prediction of Close values (TCS) in Conv1D-LSTM model on Testing Set

When the Neural Network models are built, the number of network parameters differs accordingly. Table 5.1 shows the number of trainable parameters for each NN model which are used in the experiment for TCS stock. Trainable parameters are those values of the network that keeps changing during the training.

Table 1. Trainable Parameters for each Neural Network (TCS)

NN Model	Trainable Parameters
LSTM	89,249
CNN	35,321
Conv1D-LSTM	28,929

It is observed that the CNN or Conv1D networks fit the data much faster than LSTM networks, this is due to the lesser no. of trainable parameters in these models compared to LSTM network model.

The value of MSE (Mean Square Error) is used as the loss function for models used in the experiment. The metrics used for the evaluation of the models are Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE). RMSE values obtained for TCS stock after evaluating each model is shown in Table 2.

Table 2. RMSE of different models on the test set of TCS

NN Model	RMSE
LSTM	0.0091
CNN	0.0087
Conv1D-LSTM	0.0081

After the prediction on a testing dataset for TCS stock, the inverse linear transformation is applied to both the actual (y_test) and predicted values. Then MAE & MAPE values are computed on those values using Eq. 1 & 2 given in Sec. III (A). MAE and MAPE values obtained for each NN model are depicted in Table 3.

Table 3. Error and Error Percentage on TCS

NN Model	MAE (E_{MAE})	MAPE (E_{MAPE})
LSTM	63	2.23
CNN	61	2.16
Conv1D–LSTM	56	1.98

C. Results on MRF Dataset

Fig. 14, 15, 16 represents the plots of epochs vs. loss for CNN, LSTM & Conv1D-LSTM network represented in Sec. III (B). The validation loss found to be less than the training loss at some initial epochs because it is calculated at the end of each epoch and before the start of a new epoch when the model is not optimizing.

Unlike TCS, the validation loss does not become less than the training loss when models were fitted on MRF data.

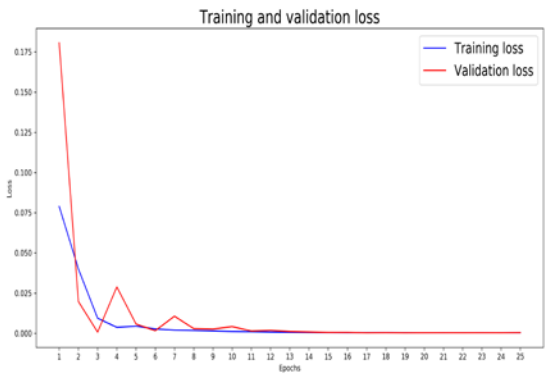

Fig. 14 Training and Validation Loss of CNN Model (MRF Data).

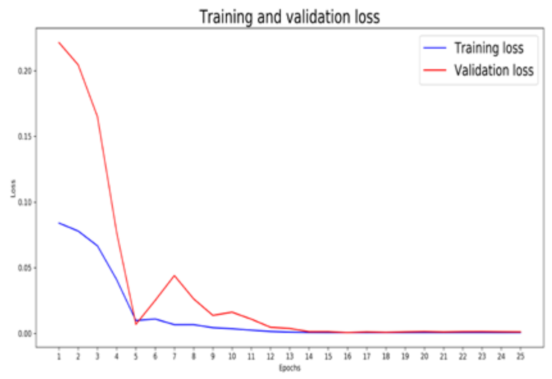

Fig.15 Training and Validation Loss of LSTM Model (MRF Data).

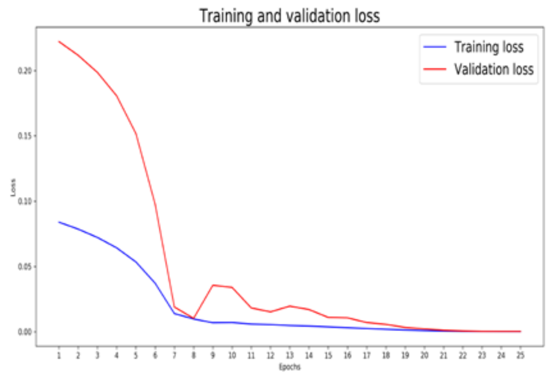

Fig.16 Training and Validation Loss of Conv1D-LSTM Model (MRF Data).

The plots obtained when close values are predicted on the Training set of MRF containing 1060 samples are shown in

Fig. 17-19 for CNN, LSTM & Conv1D-LSTM Network respectively. Although, all the networks were able to fit on training samples represented by a red line in figures. But it is evident that the CNN model in fig. 17 is able to capture the dynamics of time-data better than the Conv1D_LSTM model in fig.19 and LSTM model in fig.18 in case of MRF stock. The LSTM model is unable to validate samples properly present in the validation samples as represented by a magenta line in Fig. 18, also for some initial training samples in the training set of MRF, the Conv1D-LSTM network is unable to fit the data, unlike CNN.

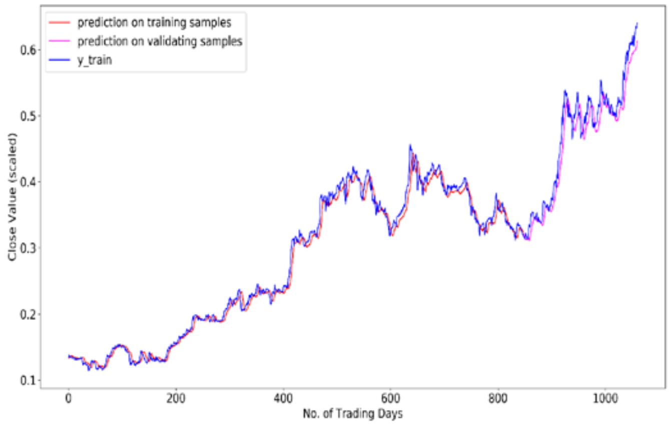

Fig.17 Close values of MRF in CNN model after 25 epochs.

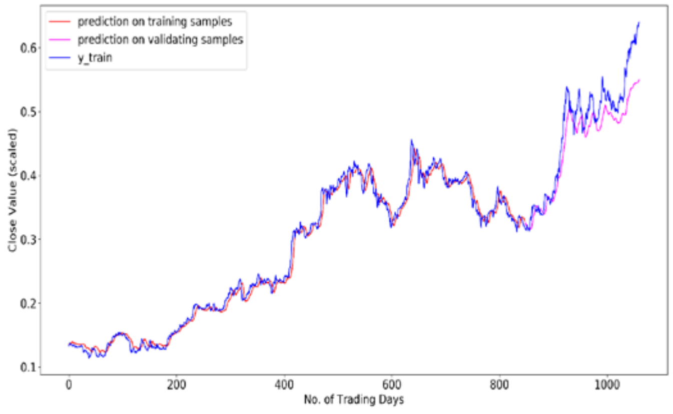

Fig.18 Close values of MRF in LSTM model after 25 epochs.

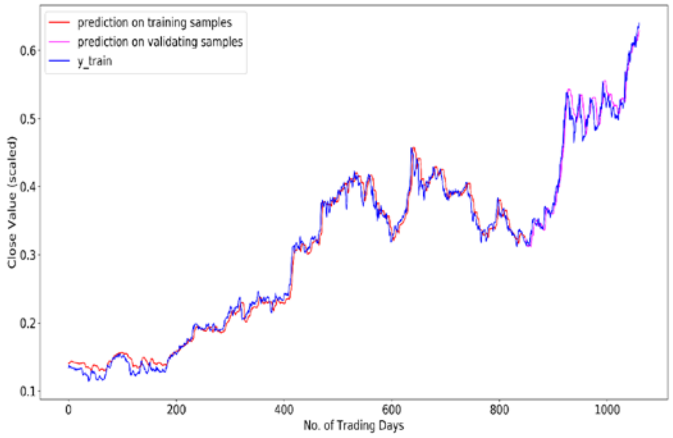

Fig.19 Close values of MRF in Conv1D-LSTM model after 25 epochs

The plots obtained when close values are predicted on testing set of MRF data containing 265 samples are shown in Fig. 20 -22 for CNN, LSTM & Conv1D-LSTM network. The Predicted values are seen to closely follow the actual values which are clearly visible in the plots. Thus, the models are adept enough to follow the direction or the trend of the stock. It can be observed from the Fig.20 & 22 that CNN and Conv1D-LSTM are more capable of capturing the spatiotemporal content of the stock then LSTM, as the LSTM plot of prediction is farther than that of a plot of other two (CNN and Conv1D-LSTM) as shown in Fig.21.

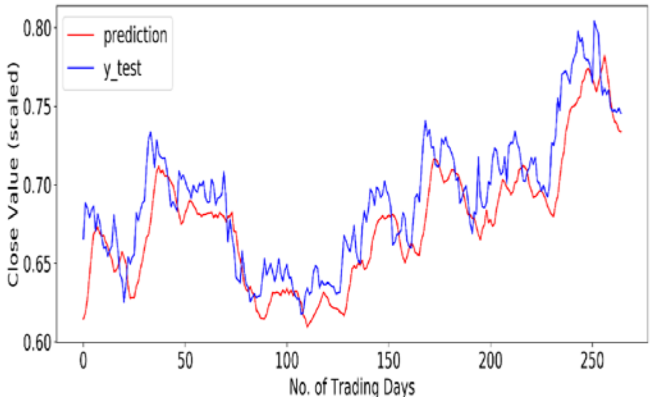

Fig.20 Prediction of Close values (MRF) in CNN model on Testing Set

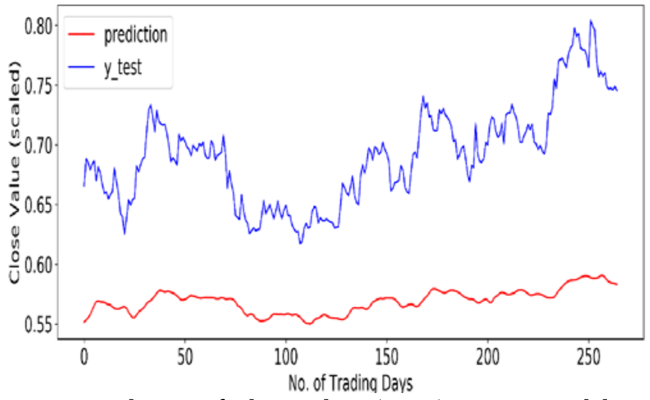

Fig.21 Prediction of Close values (MRF) in CNN model on Testing Set

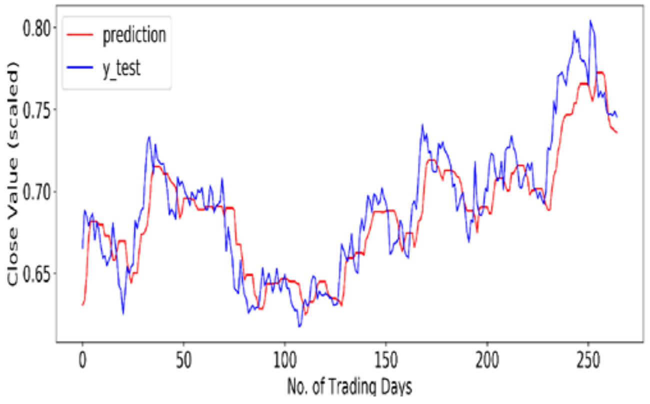

Fig.22 Prediction of Close values (MRF) in Conv1D-LSTM model on Testing Set

Table 4 shows the number of trainable parameters for each NN model which are used in the experiment for MRF stock.

Trainable parameters are those values of the network that keeps changing during the training.

Table 4 Trainable Parameters for each Neural Network (MRF)

NN Model	Trainable Parameters
LSTM	89,249
CNN	35,321
Conv1D-LSTM	29,609

It is observed that the CNN or Conv1D networks fit the data much faster than LSTM networks for MRF stock which is similar when these neural networks were used for TCS stock. This is due to the lesser no. of trainable parameters in these models compared to LSTM network model.

RMSE values obtained after evaluating each model for MRF stock is shown in Table 5.

Table 5 RMSE of different models on the test set of MRF stock

NN Model	RMSE
LSTM	0.1364
CNN	0.0226
Conv1D - LSTM	0.0179

After the prediction on a testing dataset of MRF stock, the inverse linear transformation is applied to both the actual (y_test) and predicted values. Then MAE & MAPE values are computed on those values using Eq. 1 & 2 given in Sec. III (A). MAE and MAPE values obtained for each NN model are depicted in Table 6.

Table 5.6 Error and Error Percentage on MRF

NN Model	MAE (E_{MAE})	MAPE (E_{MAPE})
CNN	1774	18.86
LSTM	13193	2.539
Conv1D – LSTM	1402	2.021

From Table 2, 3 of TCS and Table 5, 6 of MRF, it is evident that the 1D Convolutional network either used independently as CNN or used in combination with other network layers like LSTM as Conv1D-LSTM gives better performance than the LSTM network. Reduction in RMSE in case of Conv1D-LSTM compared to LSTM and CNN justifies the statement. Also, the mean absolute error and the mean absolute percentage error are found to be low when used in combination model compare to other models.

V. Conclusion

A. Conclusion

The proposed work involves the use of deep neural network models like CNN & LSTM for the short-term prediction on stock data and its comparison with the Conv1D-LSTM deep neural network. Daily stock data of TCS and MRF was used for short-term prediction. Open, high, low, close values of stock are used for analysis. Prediction is done using data of previous 5 trading days sequentially.

From the results, it is evident that NN models like CNN, LSTM and Conv1D-LSTM can be employed for short-term prediction of stock data. Presence of the large number of network parameters in LSTM network makes it computationally expensive. Although LSTM network is very good and consistent for problems based on sequence and time, the CNN and Conv1D-LSTM networks provide a faster and cheaper alternative to LSTM network for stock price prediction problem. It is observed that by using layers of different models to obtain Conv1D-LSTM deep neural network [14] gives precise results in terms of RMSE, MAE and MAPE.

RMSE, MAE & MAPE values obtained with Conv1D-LSTM Network for TCS stock data are found to be 0.0081, 56, 1.98 respectively which are much low compared to CNN and LSTM network. Similarly, for MRF stock data values of RMSE, MAE & MAPE obtained with Con1D-LSTM Network are found to be 0.0179, 1402, 2.021 respectively. These results were obtained after the training the DNNs for 35 epochs on TCS data and 25 epochs on MRF data with batch size as 128. With the use of combination models like Conv1D-LSTM, both the spatial and temporal implication of a stock price problem can be dealt easily.

The reduction in values of RMSE, MAE and MAPE with Conv1D-LSTM network compared to those obtained with CNN and LSTM indicate that combination model can fit the data more efficiently. Also, it is found that with use of Conv1D-LSTM network convergence can be reached in a lesser number of epochs. In general, Conv1D-LSTM network is efficient for short-term stock price prediction but depending on nature of stock hyper-parameters may require some variations.

B. Future Work

The inputs to the models can be considered as minimum input for solving the problem of stock price prediction, which is the reason why models in this experiment can be updated with other stock indices in future for better prediction. The models used in the experiment are not yet generalized for stocks and hence, the models can be optimized using hyper-parameter optimization. The models in the experiment have solved the problem as a regression. Regression model can be combined with Classification model (i.e. those models which are only predicting rise and fall.) to improve the directional accuracy for precise predictions.

References

[1] Jürgen Schmidhuber, “Deep learning in neural networks: An overview”, Neural Networks, Volume 61, 2015, Pages 85-117.

[2] Hochreiter, Sepp, and Jürgen Schmidhuber― Long Short-term memory, Neural Computation 9.8, 1997, pp. 1735-1780.

[3] Y. LeCunn and Y. Bengio. “Convolutional networks for images, speech, and time-series” In M. A. Arbib, editor, The Handbook of Brain Theory and Neural Networks. MIT Press, 1995.

[4] Di Persio, Luca & Honchar, O. “Artificial neural networks architectures for stock price prediction: Comparisons and applications.” International Journal of Circuits, Systems and Signal Processing Volume 10, 2016, pp .403-413

[5] Selvin, Sreelekshmy “Stock price prediction using LSTM, RNN and CNN-sliding window model.” International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2017, 1643-1647.

[6] Wei Huang, Yoshiteru Nakamori, Shou-Yang Wang, “Forecasting stock market movement direction with support vector machine”, Computers and Operations Research, archive Volume 32, Issue 10, 2005, pp. 2513–2522.

[7] [5] Marcelo S. Lauretto, B. C. Silva and P. M. Andrade, “Evaluation of a Supervised Learning Approach for Stock Market Operations”, arXiv: 1301.4944 [Stat.ML], 2013

[8] Ayman E. Khedr, S.E.Salama, Nagwa Yaseen, "Predicting Stock Market Behavior using Data Mining Technique and News Sentiment Analysis", International Journal of Intelligent Systems and Applications(IJISA), Vol.9, No.7, pp.22-30, 2017.

[9] D. Ashok Kumar and S. Murugan, “Performance Analysis of Indian Stock Market Index using Neural Network Time Series Model”, International Conference on Pattern Recognition, Informatics and Mobile Engineering (PRIME), IEEE, 978-1-4673-5845-3, 2013.

[10] Le Roux N, Y. Bengio, “Deep belief networks are compact universal approximators”. Neural Computation 22, pp. 2192–2207, 2010.

[11] F. Chollet “Deep Learning with Python” , Manning Publications Co. , Chap 6 , 2018, pp. 225-232

[12] Quandl TCS Dataset on link: https://www.quandl.com/data/NSE/TCS-Tata-Consultancy-Services-Limited, MRF Dataset on link https://www.quandl.com/data/NSE/MRF-MRF-Limited.

[13] Kingma, Diederik P.; Ba, Jimmy,” Adam: A Method for Stochastic Optimization”, CoRR, abs/1412.6980, 2014.

[14] F. Karim1, S. Majumdar, “LSTM Fully Convolutional Networks for Time Series Classification”, arXiv:1709.05206v1, 2017.