Forecasting variance of NiftyIT index with RNN and DNN

C R Karthik1, Raghunandan1, B Ashwath Rao1, N V Subba Reddy1

1 Department of Computer Science and Engineering,
Manipal Institute of Technology,
Manipal Academy Of Higher Education,
Manipal, Karnataka, India-576104

karthikcr604@gmail.com, raghun853@gmail.com, ashwath.rao.b@gmail.com, nvs.reddy@manipal.edu

Abstract. A time series is an order of observations engaged serially in time. The prime objective of time series analysis is to build mathematical models that provide reasonable descriptions from training data. The goal of time series analysis is to forecast the forthcoming values of a series based on the history of the same series. Forecasting of stock markets is a thought-provoking problem because of the number of possible variables as well as volatile noise that may contribute to the prices of the stock. However, the capability to analyze stock market leanings could be vital to investors, traders and researchers, hence has been of continued interest. Plentiful arithmetical and machine learning practices have been discovered for stock analysis and forecasting/prediction. In this paper, we perform a comparative study on two very capable artificial neural network models i) Deep Neural Network (DNN) and ii) Long Short-Term Memory (LSTM) a type of recurrent neural network (RNN) in predicting the daily variance of NIFTYIT in BSE (Bombay Stock Exchange) and NSE (National Stock Exchange) markets. DNN was chosen due to its capability to handle complex data with substantial performance and better generalization without being saturated. LSTM model was decided, as it contains intermediary memory which can hold the historic patterns and occurrence of the next prediction depends on the values that preceded it. With both networks, measures were taken to reduce overfitting. Daily predictions of the NIFTYIT index were made to test the generalizability of the models. Both networks performed well at making daily predictions, and both generalized admirably to make daily predictions of the NiftyIT data. The LSTM-RNN outpaced the DNN in terms of forecasting and thus, grips more potential for making longer-term estimates.

1. Introduction

Time-series models are very beneficial models when serially and linearly correlated data has to be analyzed. High probability businesses toil on time-series data to analyze

- Stock Market predicting/Analysis
- Website Traffic management
- Budget Analysis
- Demand for certain products
- Sales statistics for the upcoming year
- Competition Situation
- Census Analysis

But time-series are more thought-provoking and challenging of these 2 parameters: Time Dependence - The elementary assumption of the observations are independent is not true in the case of time series. Seasonality - Laterally with a decreasing or an increasing movement, most of the time-series situations have an approximate form of seasonal trends. That is said to be the variance precise to a certain frame of time.
Initially, statistical models such as and linear regression the ARIMA (Auto-Regressive Integrated Moving Average) were used for forecasting prices of stock. Researchers have built various artificial neural network (NN) models for the analysis of time series data which is linearly dependent on time. NN is the model which connects abundant artificial neurons (each neuron accomplishing basic computation) organized to generate a compound model which learns to extract features, recognize trends in the data, and depict data in a compressed format.

![Typical ANN](image1)

Figure 1. Typical ANN

Artificial Neural Network principally contains three different layers:
- **Input layer**: Accepts input data of various formats from the user of the model. **Hidden layer**: It stands in between input and output layers. It accomplishes the necessary calculations to identify hidden patterns and features. **Output layer**: The ANN receipts input and calculates the weighted sum of the input. This calculation is shown in the form of a transfer function. Transfer function determines weighted total is provided as an input to an activation function to achieve the output. Activation functions elect whether a node should be accepted or not. Those nodes which are fired are considered to the output layer. Based on the sort of task that is being performed various types of distinct activation functions can be chosen.

![A Multi-layered Neural Network](image2)

Figure 2. A Multi-layered Neural Network

Deep neural networks (DNN) are a type of neural network which consists of more than one hidden layer of neurons. The implanted layers in a DNN allow it to denote data with a high level of abstraction, depending on the number of layers. Recurrent neural networks (RNNs) are an influential and robust type of neural network and fit into the most capable algorithms as it comprises internal memory. An RNN has two kinds of inputs: current and recent past. This part of RNN is extremely important as it holds series of data in memory. This feature of RNN is what it makes distinctive from other algorithmic models.
Nifty IT index encompasses top ten corporations that derive their turnover from information technology-related actions like IT education, IT Infrastructure, Telecommunication Service, Software training, Software Development, Networking Infrastructure, Vending, Hardware manufacturers, Support, and maintenance. Nifty-IT measures the performance of India’s BSE IT stocks.

Figure 3. NIFTY IT Companies list.

The ability to analyze stocks and their market tendencies could be precious to traders and investors, and thus there is sustained interest. Abundant statistical and machine learning methods are already explored for stock prediction, analysis, and forecast. In this paper, we present a comparative study of two very promising artificial neural network models namely a Long Short-Term Memory (LSTM) recurrent neural network (RNN) and a deep neural network (DNN) in forecasting the daily movements of the Indian NiftyIT index.

The novelty in our work is the combination of NiftyIT Dataset and the DNN-LSTM models applied. As most of the papers predict data of only a single stock with a limited timeframe. Our approach contains a wider time window along with the aggregate data of NifityIT (10 companies).

2. Literature Survey
Stock price prediction is a well-known problem that has been studied in various ways. Statistical methods are used like Linear Regression and Support Vector Machine (SVM) by Panwar et. al.[1] where they have applied the methods on web scraped dataset, from before 2008 which has little variation and hence, not a very good representation of the current stock market. A subsection Some text.

Ariyo et. al. [2] have used the autoregressive integrated moving average (ARIMA) model on New York Stock Exchange (NYSE) and Nigeria Stock Exchange (NSE) datasets. The models performed well for short-term prediction and have shown promise when compared with the existing methods. The paper by Zihao Gao [3] provides a comparative study of LSTM and Seq2Seq models with and without attention layer. The Seq2Seq model with attention was shown to have a demonstrable low latency output which is desirable in real-life stock price prediction systems.
The usage of Artificial Neural Network (ANN) was shown by Xuefei Kan et.al. [4] and there is a case to be made regarding the use of denser ANN called Deep Neural Network (DNN) as showcased by [5][6][7]. Yilin Ma et. al. [5] have tried to provide a solution to the portfolio optimization problem and have compared the models based on DNN, LSTM, and CNN. They applied the models for prediction and the resultant error metrics were used to make a risk assessment on the portfolio predicted by the models. They concluded that DNN gave the best returns for portfolio management.

Mohamed Al Arad et. al. in [6] demonstrated the usage of a hybrid method that makes use of both DNN and LSTM for predicting the direction of the stock market i.e. a daily label of the direction up or down for the next day’s stock market. Dev Shah et.al. [7] used DNN and Long Short Term Memory (LSTM) to forecast the Daily and weekly predictions on the Tech Mahindra stock. They conducted multiple experiments by changing parameters such as number of hidden layers, number of neurons, etc., and proposed an optimum design for the DNN and LSTM. They reported that while DNN performed well for the daily predictions, the LSTM performed better in the longer-term predictions, this was credited to the fact that the LSTM has memory to ‘remember’ the values and are not a point prediction method like the DNN.

Dou Wei in [8] proposed LSTMs with an additional layer called an attention layer that would learn based on the weights in the previous layers. It performed better for a dataset of 2 years rather than 18 month’s data of Shenzhen stock index and HS300 index which could be attributed to the nature of the Deep Learning network which works better on larger datasets. Mohammad Asiful Hossain et. al. [9] used a hybrid approach involving LSTM and Gated Recurrent Unit (GRU). They provided the first level prediction output from the LSTM layer to GRU Layer to perform the final predictions. This produced results with significantly reduced Mean Square Errors.

Sidra Mehtab et. al. in [10] investigated performance of LSTM models on the NIFTY 50 index. They considered the daily historic data of the index. They found that LSTM models provided superior performance when compared with the other Machine learning models.

3. Dataset
A dataset is a collection of data fragments that can be treated by a processor as a solitary unit for analysis and estimate purposes. Data set of NiftyIT index for the past 10 years have been retrieved from BSE (Bombay stock exchange) official website [15]. To maintain standardization of data we have retrieved from the official website. Features consider in our data set are Date, open price, high, low, close price, and Volume of shares traded on a particular date in the market. We have zeroed in 90 % data for training purposes and 30% of data in the same dataset is considered for testing the LSTM and DNN machine learning models.

4. Methodology

4.1. Deep Neural Network (DNN)
They are a class of Artificial Neural Networks that have multiple hidden layers and can use backpropagation algorithms to learn the patterns from the dataset, Figure 4 shows a representation of a DNN with three hidden layers.
The basic structure of the Neuron (also known as perceptron) is shown in Figure 5.

It has an input layer, the number of neurons here is equal to the number of features in the dataset. The inputs are scaled to be values between 0 and 1, they are multiplied with the corresponding weights and biases added before they are passed on to the next layer, this equation is shown in Equation 1. In the next layer, the inputs are summed as given in Equation 2 and are then passed on to a suitable Activation function. The Activation functions are listed in Table 1.

\[ y = WX + \text{bias} \]  
(1)

\[ x_{in} = \sum_{k=0}^{n} w_i x_i + b_i \]  
(2)

Figure 4. A Deep Neural Network. [11]

Figure 5: A Neuron used in DNN. [12]
Table 1: List of activation functions. [13]

| Name                  | Plot | Equation                      | Derivative       |
|-----------------------|------|-------------------------------|------------------|
| Identity              |      | \( f(x) = x \)               | \( f'(x) = 1 \)  |
| Binary step           |      | \( f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases} \) | \( f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases} \) |
| Logistic (s.l.s) Soft step | | \( f(x) = \frac{1}{1 + e^{-x}} \) | \( f'(x) = f(x)(1 - f(x)) \) |
| TanH                  |      | \( f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \) | \( f'(x) = 1 - f(x)^2 \) |
| ArcTan                |      | \( f(x) = \tan^{-1}(x) \)   | \( f'(x) = \frac{1}{x^2 + 1} \) |
| Rectified Linear Unit (ReLU) | | \( f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases} \) | \( f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases} \) |
| Parametric Rectified Linear Unit (PReLU) | | \( f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases} \) | \( f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases} \) |
| Exponential Linear Unit (ELU) | | \( f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases} \) | \( f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases} \) |
| SoftPlus              |      | \( f(x) = \log_e(1 + e^x) \) | \( f'(x) = \frac{1}{1 + e^{-x}} \) |

The Output layer is limited to using only the Sigmoid function for the activation as it is a smooth function that gives the values between 0 and 1. As shown in graph Figure 6.

![Sigmoid Function and graph.](image)

Figure 6: Sigmoid function and graph. [14]

4.1.1. DNN Code Implementation

The DNN architecture initially consisted of one hidden layer of 30 Neurons with Rectified Linear Unit (ReLU) activation and 50 epochs of training. After extensive testing of multiple different architectures, the architecture of 3 hidden layers with 20, 18, and 21 Neurons respectively was chosen and an epoch number of 30 was chosen arbitrarily to reduce the learning time without affecting the
learning loss too much. The predictions were done daily for 10% of the data and the metrics were captured.

The DNN was implemented using Keras Library in python which uses TensorFlow in the backend. ‘Adam’ optimizer was used along with ‘mean squared error’ loss to track the training loss of the model. The model training was stopped when there was no improvement in the loss metric for 3 epochs this was achieved by setting the early stopping patience metric to a value of 3 during the training. Figure 4 shows the code flow chart.

4.1.2. DNN Algorithmic Steps.
The Algorithmic steps for the DNN code are as follows
Step 1: Get the data from the CSV file.
Step 2: Pre-process the data namely cleaning and scaling.
Step 3: Perform test-train split of the dataset in 90:10
Step 4: Define and build the model as 2-20-18-21-1 with Relu as the hidden layer activation function
Step 5: Compile the model
Step 6: Evaluate the model and capture the metrics
Step 7: Inverse transform the predicted output and plot the graphs for further processing

![Figure 7: Code flow chart.](image1)

![Figure 7: Code flow chart.](image2)

4.2. Long Short Term Memory (LSTM)
Long Short Term Memory – LSTM is a form of Recurrent neural network (RNN). In RNN output from the previous phase is provided as input to the current phase. LSTM solved the problem of storing long term as RNN was precisely good for predicting data in short term memory. Hence, RNN did not perform well for elongated time series data whereas LSTM could perform it well. LSTM is used for predicting, processing, and analyzing data that are linearly dependent on time. Structure of LSTM: It
has a chain construction that encompasses 4 neural networks (NN) and varied blocks of memory called cells.

Information is held in reserve by the cells and the memory actions are done through the gates. There are 3 gates – **Forget Gate**: The data that is no longer beneficial in the cell state is detached from the forget gate. **Input Gate**: Accumulation of valuable information to cell state is performed by the input gate. **Output Gate**: The mission of mining valuable information from the current cell state to be obtainable as output is performed by the output gate.

### 4.2.1. LSTM Code Implementation

Level 0 DFD gives overview information of the whole system or single high-level process. Level 1 DFD highlights the main sub-process of a system considering it as a single unit.
The LSTM architecture initially consisted of one hidden layer of 50 Neurons with Rectified tanh activation and 20 epochs of training. The LSTM was implemented using Keras Library in python which uses TensorFlow in the backend.

4.2.2. **LSTM Algorithmic Steps.**
- Step 1: Load the dataset and define the target variable
- Step 2: Cleaning and Normalizing data
- Step 3: Creating Data Frame
- Step 4: Creating train and test sets of 9:1 split
- Step 5: Create and Fit LSTM network
- Step 6: Predict values and capture metrics
- Step 7: Inverse transform the output and Plot comprehensive graph.

![LSTM Flowchart](image)

Figure 15. LSTM Flowchart

5. **Results and Analysis**

**Loss:** It is prediction error of artificial neural network. Calculating the loss is called Loss Function.

**MAE:** It measures the difference between any two continuous variables.

\[
\text{Prediction Error} = \text{Actual Value} - \text{Predicted Value}.
\]

\[
\text{Absolute Error} = |\text{Prediction Error}|.
\]

\[
\text{MAE} = \text{Average of All absolute errors}
\]

**MSE:** The Mean Squared Error is most common loss function used. To calculate the MSE, difference between model's predictions and real values are taken and squared, this is averaged out across the dataset.

**MAPE:** The mean absolute percentage error is a measure of accuracy a prediction. Here accuracy is measured as a percentage of deviation from expected result/original value.

**Cosine proximity:** Cosine proximity is a measure of similarity between predicted and expected values. A higher value of cosine proximity signifies higher accuracy. Opposite values have a cosine proximity of -1, whereas identical vectors have a cosine proximity of 1.
5.1. DNN Results and Analysis
DNN was implemented and tested on the 10 years BSE NiftyIT Dataset, we arrived at the error parameter ranges as shown in Table 2.

| Metric         | Min             | Max             |
|----------------|-----------------|-----------------|
| loss           | 9.65717263170518e-05 | 0.22650127112865448 |
| mse            | 9.65717263170518e-05 | 0.22650127112865448 |
| mae            | 0.006955412682145834 | 0.3524141013622284 |
| mape           | 5330.7373046875 | 27489.994140625 |
| cosine_proximity | 0.5277231335639954 | 0.9996067881584167 |

For the above values, the graphs have been plotted in Figures 16.

Figure 16. Error Metrics DNN

Figure 17 shows the comparison of predicted values with the original dataset values. The predicted values have been superimposed on the original ‘close’ values for the test data (10% of the complete dataset). The superimposed predicted values from the DNN model are with higher accuracy and lesser loss metrics. This is as seen from the figure 16 and 17.
5.2. LSTM Results and Analysis

LSTM was implemented and tested on the 10 years BSE NiftyIT Dataset, we arrived at the error parameter ranges as shown in Table 3.

Table 3. Error Metrics and their ranges for LSTM

| Metric       | Min                               | Max                               |
|--------------|-----------------------------------|-----------------------------------|
| loss         | 8.233643893618137e-05             | 0.0003974180726800114             |
| mse          | 8.233643893618137e-05             | 0.0003974180726800114             |
| mae          | 0.006587174255400896              | 0.013071541674435139              |
| mape         | 966.6782836914062                 | 7361.17431640625                  |
| cosine_proximity | 0.999597251415257             | 0.9955698847770691                |

For the above values, the graphs have been plotted in Figure 18.
Figure 18. Error Metrics LSTM

Figure 19 shows the comparison of predicted values with the original dataset values. The predicted values have been superimposed on the original 'close' values for the test data (10% of the complete dataset). Predicted accuracy for LSTM was found to be better than the DNN as we have an extra memory component which performs better for the time series data. It is also observed that loss metrics is significantly lesser in the case LSTM.

Figure 19. LSTM predictions.
5.3. **DNN vs LSTM**

The DNN performed well in predictions, particularly in the cases where the trend of the true data was very clear. However, when there were multiple abrupt changes within true data, the DNN resisted. The LSTM-RNN was better able to distinguish and recognize the directivity of the variations in the true data.

6. **Conclusion and Future scope**

The aim of predicting NiftyIT index values was successfully achieved. NiftyIT Indexes were forecasted for 10% of original data from the BSE NiftyIT dataset where 90% was considered as the training data. We observed that both DNN and LSTM-RNN models performed acceptably well. Initially, DNN was considered for the forecasting. Later, LSTM was incorporated due to DNN's point prediction nature. As a result of the above action, we witnessed better accuracy for the LSTM model.

As a future scope, we intend to dynamically provide the prediction on day to day basis in real-time as an improvement to the static prediction model which is being used currently. Also, we anticipate a need for a platform that accepts the stock name from the user and predicts the futuristic daily closes as a response for the stock provided.

**References**

[1] B. Panwar, G. Dhuriya, P. Johri, S. Singh Yadav and N. Gaur, "Stock Market Prediction Using Linear Regression and SVM," 2021 International Conference on Advanced Computing and Innovative Technologies in Engineering (ICACITE), 2021, pp. 629-631, doi: 10.1109/ICACITE51222.2021.9404733.

[2] A. A. Ario, A. O. Adewumi and C. K. Ayo, "Stock Price Prediction Using the ARIMA Model," 2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation, 2014, pp. 106-112, DOI: 10.1109/UKSim.2014.67.

[3] Z. Gao, "Stock Price Prediction With ARIMA and Deep Learning Models," 2021 IEEE 6th International Conference on Big Data Analytics (ICBDA), 2021, pp. 61-68, doi: 10.1109/ICBDA51983.2021.9403037.

[4] X. Kan, M. Miao, L. Cao, T. Xu, Y. Li, and J. Jiang, "Stock Price Prediction Based on Artificial Neural Network," 2020 2nd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI), 2020, pp. 182-185, doi: 10.1109/MLBDBI51377.2020.00040.

[5] Y. Ma, R. Han, and W. Wang, "Prediction-Based Portfolio Optimization Models Using Deep Neural Networks," in IEEE Access, vol. 8, pp. 115393-115405, 2020, doi: 10.1109/ACCESS.2020.3003819.

[6] M. Al Aradi and N. Hewahi, "Prediction of Stock Price and Direction Using Neural Networks: Datasets Hybrid Modeling Approach," 2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI), 2020, pp. 1-6, doi: 10.1109/ICDABI51230.2020.9325697.

[7] D. Shah, W. Campbell, and F. H. Zulkernine, "A Comparative Study of LSTM and DNN for Stock Market Forecasting," 2018 IEEE International Conference on Big Data (Big Data), 2018, pp. 4148-4155, doi: 10.1109/BigData.2018.8622462.

[8] D. Wei, "Prediction of Stock Price Based on LSTM Neural Network," 2019 International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM), 2019, pp. 544-547, doi: 10.1109/AIAM48774.2019.00113.

[9] M. A. Hosseini, R. Karim, R. Thulasiram, N. D. B. Bruce and Y. Wang, "Hybrid Deep Learning Model for Stock Price Prediction," 2018 IEEE Symposium Series on Computational Intelligence (SSCI), 2018, pp. 1837-1844, doi: 10.1109/SSCI.2018.8628641.

[10] J. Sen, S. Mehtaband A. Dutta, "Stock Price Prediction Using Machine Learning and LSTM-
Based Deep Learning Models”. TechRxiv, 05-Aug-2021, doi: 10.36227/techrxiv.15103602.v1.

[11] Ravindra Parmar, “Training Deep Neural Networks”, Towards Data Science, https://towardsdatascience.com/training-deep-neural-networks-9fdb1964b964, accessed on 27 Jun. 2021.

[12] Nick Winovich, “Deep Learning”, Jan 2019, https://www.math.purdue.edu/~nwinovic/deep_learning.html, accessed on 27 Jun. 2021.

[13] Sagar Sharma, “Activation Functions in Neural Networks”, September 2017, https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6, accessed on 27 Jun. 2021.

[14] Ayush Pant “Introduction to Logistic Regression”, Jan. 2019, https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148, accessed on 27 Jun. 2021.

[15] BSE Official - https://www.bseindia.com/, accessed on 27 Jun. 2021.