International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-9, Issue-7, May 2020

Predicting Stock Market Prices using Fine-Tuned IndRNN

Ahmad Alsharef, Prachet Bhuyan, Abhishek Ray

Abstract*: Prediction and analysis of stock market data have a vital role in current time's economy. The various methods used for the prediction can be classified into 1) Linear Algorithms like Moving Average (MA) and Auto-Regressive Integrated Moving Average (ARIMA). 2) Non-Linear Models like Artificial Neural Networks and Deep Learning. In this work, we are using the results of previous research papers to demonstrate the potential of some models like ARIMA, Multi-Layer Perception (MLP)), Convolutional Neural Neural Network (CNN), Recurrent Neural Network (RNN), Gated Recurrent Unit (GRU), Long-Short Term Memory (LSTM) for forecasting the stock price of an organization based on its available historical data. Then, implementing some of these methods to check and compare their efficiency within the same issue. We used Independently RNN (IndRNN) to explore a better efficiency for stock prediction and we found that it gives better accuracy prevailing methods in the current time. We also proposed an enhancement to IndRNN by replacing its default activation function with a more effective function called Parametric Rectified Linear Unit (PreLU). Our proposed approach can be used as an alternative method for predicting time series data efficiently other than the typical approaches today.

Keywords*: Deep learning, Independently Recurrent Neural Network, Parametric rectified linear unit, Stock prediction, Time-series forecasting.

I. INTRODUCTION

The stock exchange is among the crucial components of the free market economy as it provides firms to evaluate their capital wherein the intention of any investor as soon as he decides to shop for or promote shares is income maximization. We worked with historical information regarding the stock prices of a publicly indexed company. We implemented a combination of simple and advanced algorithms to predict the stock price of the Auto-Mobile sector. This venture intends to predict the following days' stocks returns using IndRNN. The algorithms tested in this research included MA, ARIMA, LSTM, and IndRNN which is an updated version of RNN. Finally, we fine-tuned the IndRNN itself for better performance and accuracy.

Revised Manuscript Received on May 30, 2020.
***** Correspondence Author
Ahmad Alsharef*, School fo Computer Engineering, Kalinga Institute of Industrial Technology Deemed to be University/ Bhubaneswar, India.
Prachet Bhuyan, School fo Computer Engineering, Kalinga Institute of Industrial Technology Deemed to be University/ Bhubaneswar, India.
Abhishek Ray, School fo Computer Engineering, Kalinga Institute of Industrial Technology Deemed to be University/ Bhubaneswar, India.

II. LITERATURE SURVEY

A. Linear Prediction Algorithms

1) Moving average (MA):

It is a method that creates a series of averages of different subsets of the complete dataset, commonly used with time-series data. It can be used in technical analysis of financial data, like stock prices, returns or trading volumes. An example of using MA is when predicting the next day's stock price depending on the previous n-day sample of the closing price is the mean of the previous n days' closing prices [6]. If those prices were $P_1, P_2, \ldots\ldots, P_{n-1}, P_n$ then the next day's price is: $P_{n+1} = \frac{P_1 + P_2 + \cdots + P_{n-1} + P_n}{n}$ [9].

2) Auto-regressive integrated moving average (ARIMA):

A linear prediction algorithm considers that the past values of the time-series data can unaccompanied be employed in predicting the future values of the same data [6].

The AR (Auto-Regressive) part of ARIMA shows that the future value Y_t depends on its past values $Y_{t-3}, Y_{t-2}, Y_{t-1}$, etc. The common representation of AR(P) is:

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \cdots + \beta_p Y_{t-p} + \varepsilon_t$$

Where **ε** is the random error and parameter p is the number of the previous periods to consider in predicting. In our case, the number of previous days to consider when predicting the next day price.

The I (Integrated) part represents that the data values are replaced by the difference between the data values and this differentiating process may have been performed more than once. We use the letter d to represent the parameter that represents the number of these differences [3].

The MA (Moving Average) part indicates that the future value Y_t depends on the random error terms. A common representation of MA(q) is:

$$Y_t = \beta_0 + \varepsilon_t + \Phi_1 \varepsilon_{t-1} + \Phi_2 \varepsilon_{t-2} + \cdots + \Phi_q \varepsilon_{t-q}$$

Where q is the number of error terms to consider in predicting.

ARIMA full formula is given as:

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + .. + \beta_p Y_{t-p} + \Phi_1 \varepsilon_{t-1} + \Phi_2 \varepsilon_{t-2} + .. + \Phi_q \varepsilon_{t-q}$$

ARIMA configurations are represented in the form ARIMA(p,d,q) [11].

B. Deep Learning Algorithms

1) Multi-layer perception (MLP):

Also known as feed-forward network. It has three types of layers: input, hidden, and output layers. Neurons within the same layer aren't connected but each neuron in a layer is connected to the next layer's neurons via a weighted matrix [7].

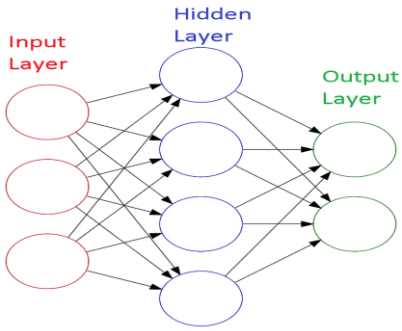

Figure 1: Multi-Layer Perception

2) Convolutional neural network (CNN):

It is a feed-forward neural network basically contains three types of layers, convolution layers, pooling layers, and fully-connected layers. It can take a 2D (or more dimensions) matrix and outputs a result after doing a sequence of operations [13].

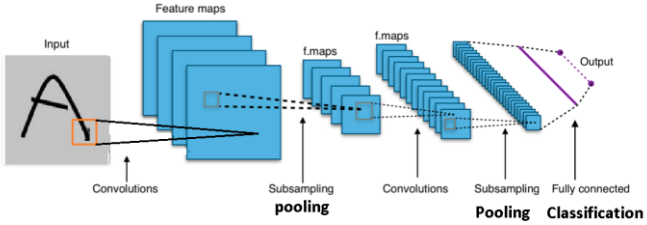

Figure 2: Convolutional Neural Network (CNN)

Convolution starts at the top left of the input image (matrix) and takes a small window of that matrix with a certain width and height and performs operations on it to get final results. It then moves the window to the right and does the same. It does the same across the whole matrix and outputs a new matrix. Pooling layers are used to combine a small window of pixels into a single pixel by running an operation on that window, usually (average, max, or min). After many convolutions and poolings, the final output is passed to a fully connected layer to output the final result [13].

3) Recurrent neural network (RNN):

It is different from MLP, it takes input from two sources, from the present and the past. We can say, it has a memory. The input information is stored as a hidden state and used recursively as it sweeps forward to deal with a new sample [7]. In an RNN, at each time step, the network performs the same function, hence, the term recurrent in its name. The architecture of an RNN is illustrated in the following diagram:

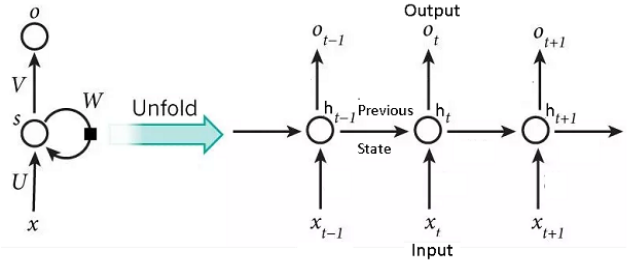

Figure 3: Recurrent Neural Network

At each time step, t, a memory state, h_t, is computed based on the previous sate h_{t-1}, at step (t-1) and the input, x_t, at time step t.There are many cases where RNNs are poorly-suited. For example, predicting the next word in a sentence when we don't only need to know the preceding context to make a strong prediction, but also the text from further back. In fact, RNN will only look at the preceding word and give a reasonably poor prediction and this is called by the problem of Long-Term Dependencies [12].

4) Gated recurrent unit (GRU):

An improved version of RNN uses two gates to decide what information should be passed to the output, an update gate and a reset gate. They can be trained to keep time-series data, without deleting it through time or removing irrelevant information [10].

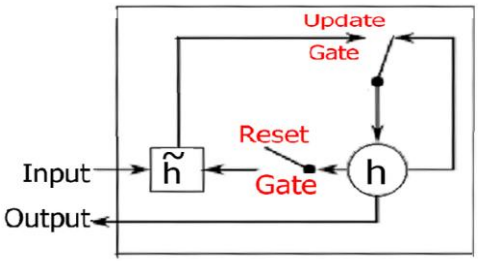

Figure 4: Gated Recurrent Unit (GRU)

5) Long short-term memory (LSTM):

It was designed to deal with long-term dependencies by remembering the information for a long time back. It contains several memory blocks or cells. Each of them has three gates and a state to adjust the data flow through the cells.In the problem of predicting the next word in a sentence, LSTMs perform better due to the memory block by remembering information of a long time back [7].

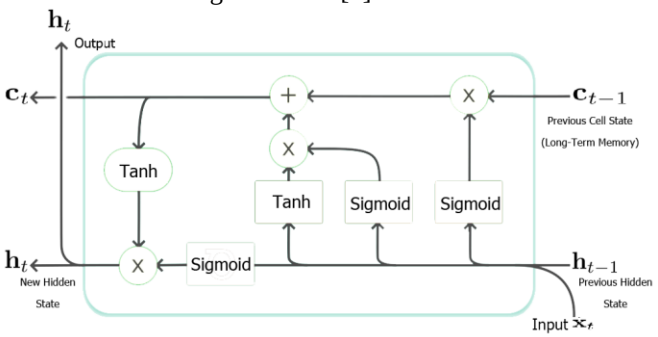

Figure 5: LSTM Cell

6) Independent recurrent neural network (IndRNN):

Shuai Li et al. (2018) [8] proposed this enhanced type of RNN. The fundamental IndRNN structure is represented in the following figure:

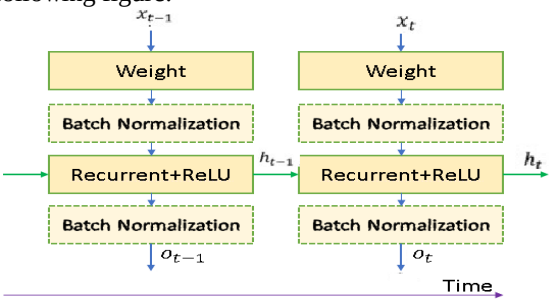

Figure 6: Basic architecture of IndRNN of Shuai Li [8]

Where Weight and Recurrent+ReLU represent the processing of the input and the recurrent procedure at each step using ReLU activation function.

Batch normalization can be done before or after the activation function as represented in the figure.A deep IndRNN network can be created by accumulating this basic architecture. Compared to LSTM which often uses Sigmoid and Tanh activation functions, ReLU non-saturated activation function which is used in IndRNNs reduces the gradient vanishing problem over layers.

IndRNN can be described as:

$h_t = \sigma(W_{x_t} + U \odot h_{t-1} + b)$

Where W_{x_t} is the input weight, U is the recurrent weight which is a vector and $\odot$ symbolizes Hadamard product. The hidden state for the n-th neuron can be calculated as:

$h_{n,t} = \sigma(W_{xn_t} X_t + U_n h_{n,t-1} + b_n)$

Where W_n is the n-th row of the input weight and U_n is the n-th row of the recurrent weight [14].

Each neuron receives data only from the input and its own hidden state at the previous time step [8].

They have shown that IndRNN can achieve better performance than a traditional RNN. Every neuron in one layer is independent of other neurons in the same layer. The connection between the neuron can be done by stacking two or more layers. Each neuron in one layer only receives its own past state as context information (instead of full connectivity to all other neurons of the previous and the next layers) and in this manner, neurons are independent of one another's history. This offers several benefits of IndRNN over the traditional RNN:

- Address the problems of gradient vanishing and exploding by regulating the gradient back-propagation through time.
- It keeps the long-term dependency to process longer sequences. Experiments show that IndRNN is able process a sequence of more than 5000-steps while LSTM is able to process a sequence of less than 1000-steps.
- It works well and can be trained robustly with a non-saturated activation function like ReLU.
- The independence of neurons in the layer makes neurons behaviors in each layer easier to interpret.
- Experiments show that IndRNN can outperform basic RNN and LSTM methods with adding problems, classification, language modeling, and sequential MNIST.

C. Gradient Vanishing and Exploding Problems

1) Gradient vanishing problem:

In many models, the neural network's weights receive updates in their values at each training iteration with respect to the current weight.

The problem is that in certain scenarios, this gradient (update) might be vanishingly small preventing the weight from changing its value. In the most unfavourable scenario, the neural network might stop further training.

For example, the range of the gradient in the classic activation functions is in range (0,1), and back-propagation calculates gradients by multiplying n of these small numbers and in the early layers of an n-layer network, we find that the gradient (update) in vanishes sharply causing the early layers to train very slowly [5].

2) Gradient exploding problem:

The exploding gradient is a problem where large gradients cause very large updates to the neural network's weights while training. This causes the model to become unstable and unable to learn from the training data [5].

D. Activation Functions

1) Rectified linear unit (ReLU):

It is the most used activation function in neural networks. It can be defined as:

$$f(x) = \max(0, x) = \begin{cases} 0; & x < 0 \\ x; & x \geq 0 \end{cases}$$

In other words, it is linear for all positive values and zeroes for all negative values [4].

The disadvantage of outputting zeros for all negative values is a problem called Dying ReLU [4] occurs when a neuron gets stuck on the negative side outputting zeros and once it gets negative values, it becomes improbable to recover. This problem is probable to take a place when the learning rate is too high or the bias is largely negative. Low learning rates often solves this problem but might result in that the training process progresses slowly as it is making too tiny updates to weights each epoch. LReLU and PReLU which are advanced activation functions solve this problem since they alternatively have a slight slope in the negative side instead of a zero slope.

2) Leaky ReLU (LReLU):

It has a tiny slope for negative values, instead of altogether zero [4]. It can be described as:

$$f(x) = \max(0, x) = \begin{cases} 0.01; & x < 0 \\ x; & x \geq 0 \end{cases}$$

3) Parametric ReLU (PreLU):

It is a type of leaky ReLU where the slopes of the negative side are alternatively learnable and not multiplied by a constant like in LReLU. The slopes in PreLU are initially randomized within a given range during training but the fixed during training [4]. It can be described as:

$$f(x) = \max(0, x) = \begin{cases} ax; & x < 0 \\ x; & x \geq 0 \end{cases}$$

where $a \sim$ U(A, B), A < B, and A, B $\in$ [0, 1].

PReLU is an adaptive Activation function. In the training phase, a is randomized from the uniform distribution U(A,B). In the test phase, it is fixed according to the average of all a parameters used in the training phase.

The performance of PReLU is investigated to be better than ReLU and LReLU in some experiments [15].

The difference between the previous activation functions can be demonstrated in the following figure:

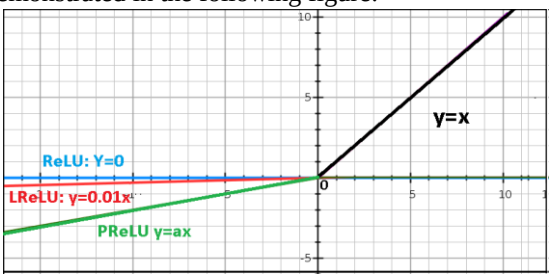

Figure 7: ReLU vs LReLU vs PReLU

Using PReLU gives many benefits over ReLU:
(i) It prevents the Dying ReLU problem, as it doesn't have zero-slopes parts [13].
(ii) PReLU offers better performance and speeds up training than standard ReLU as they do not saturate (reach the peak value either maximum peak 1 or minimum peak 0) when approaching the slope.
Unlike ReLU, leaky ReLU is more balanced, and may, therefore, learns faster [13].
(iii) It gives better accuracy when it is used with IndRNN as we proved in our research work.

III. RELATED WORK

Nishanth C P et al. (2018) [10] tried to find out the efficiency of each deep learning model when it is used to predict stock market prices. They have selected the data from the automobile and banking sectors listed on the National Stock Exchange of India (NSE).
Three different deep learning models that were discussed in their paper were RNN, LSTM, and GRU. The error percentage maximum value obtained for each of the models for both the automobile and banking sectors is represented in the following table:

Table 1:Error Percentage of Deep Learning Algorithms [10]

Company/Bank	RNN	LSTM	GRU
Tech Mahindra	7.14	6.24	6.65
TVS Motors	2.00	1.79	2.05
Syndicate Bank	1.72	1.55	1.64
Union Bank	1.57	1.34	1.40

LSTM performed better compared to the other two models in predicting both sectors' future stocks' prices. They had used 1000 epochs to train and it took a long time for obtaining the results.Hiransha M et al. (2018) [7] took datasets from three different sectors which were Automobile, IT, and Banking sectors of NSE.
The corresponding companies of these sectors were Maruti, HCL, and AXIS Bank respectively. They had used four types of neural networks: MLP, RNN, LSTM, and CNN. They also used ARIMA which is a linear model [7].
To compare these algorithms, they used Mean Absolute Percentage Error (MAPE) and the results obtained in their work were:

Table 2: MAPE using ARIMA and DL networks [7]

Company	ARIMA	RNN	LSTM	CNN	MLP
Maruti	20.66	5.82	6.03	4.05	4.81
HCL	24.69	5.40	5.52	4.40	3.58
Axis Bank	19.64	11.64	4.88	5.22	5.5

According to their results, CNN performs the best in predicting the Automobile sector, MLP performs the best with IT Sector, and LSTM performs the best with Banking Sector.Their work hasn't explored the advantages of using a hybrid network that merges two or more models in one network for prediction and the researchers have tested the algorithms for 1000 epochs which took a long time to proceed.

IV. OUR WORK

A. Workflow

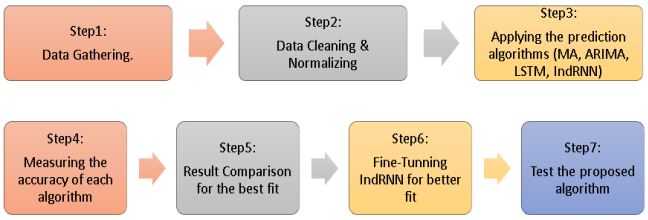

Figure 8: WorkFlow Diagram of our proposed work for predicting stock market prices

1. We started by collecting stock prices historical data of TATAMOTORS of the automobile sector of NSE from Yahoo Finance Website [2]. The data contained the daily stock prices between the periods 06-03-1996 and 01-06-2009 with 4836 observations. We considered only the close prices of each day in prediction and we used them as the input parameters of the models.
2. We cleaned the data by filling the missed values by copying the previous day value. In other words, the missed price of the stock of a particular day is the same as its price of the previous day.
3. We applied four algorithms: MA by predicting the next day price depending on the prices of the previous 10 days. ARIMA (1,0,0) = first-order autoregressive model: the series is calculated as a multiple of its own previous value, plus a constant. The forecasting equation, in this case, is $Y_t = \mu + \Phi_1 Y_{t-1}$
   LSTM, and IndRNN by predicting the next-10-days prices depending on the prices of the previous 40 days as follows: Given (t-39),... (t) → Predict (t+1),... (t+10). We divided the data into a Train dataset and a Test dataset. The training dataset contained 67% of the data and the testing dataset contained 33% of the data. In total, the Training dataset contained 3240 observations and the Testing dataset contained 1596 observations.
   The hyperparameters used in the LSTM and IndRNN neural networks were:

Table 3: Hyperparameters used with LSTM and IndRNN networks

Hyperparameter	Value	Value
Deep Learning Model	LSTM	IndRNN
Timesteps	40	40
Optimizer	Adam	Adam
Learning rate	0.001	0.001
Batch size	40	40
Epochs	200	200
Activation function(s)	Tanh+Sigmoid	ReLU
Number of hidden layers	5	5
Neurons in the hidden layer	40	40

4. We measured the accuracy of each algorithm using the MSE method.
5. We compared the MSE of each algorithm to find the best fit.

6. We fine-tuned IndRNN by changing its activation function to PReLU instead of its default ReLU.
7. We tested the fine-tuned IndRNN and found that it gave a lower MSE than IndRNN which means better accuracy.

B. Dataset used

We have performed the analysis on an Automobile TATAMOTORS stock market data scraped from publicly available Yahoo-Finance website [1]. For this, we had used four algorithms which were MA, ARIMA, LSTM, and IndRNN. The dataset included Tata Motors Ltd.'s daily stock market prices from 1996 to 2009 [1]. The dataset contained 4836 entries with general features Date, Open, High, Low, Close, Adj Close, and Volume. We only considered the parameter Close as our models' input.

C. Result analysis

The mean squared error obtained for each of our models with our dataset is given in the table below.

Table 4: MSE with Tata Motors Dataset

Algorithm	MA	ARIMA	LSTM	IndRNN
MSE	108	55	53	46

LSTM performed better compared to the other two linear models (MA and ARIMA). IndRNN performed the best.

We contributed by testing the IndRNN algorithm which was proposed by Shuai Li et al. in 2018 [8] in forecasting financial time-series data and we found that it outperformed the other algorithms. It was the first time to explore the efficiency of IndRNN in predicting future financial data.

The graphs of MA and ARIMA observations which show the difference between the forecasted data and the real data of the years 2008 and 2009 are represented in the following figures:

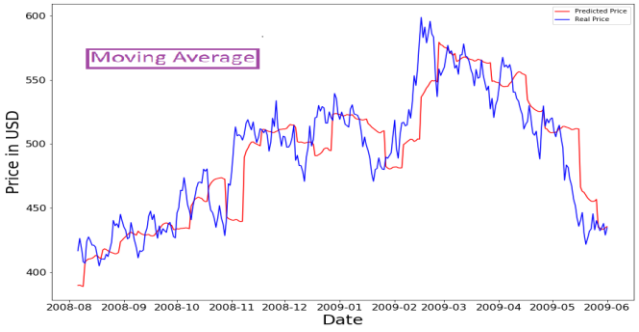

Figure 9: MA Line Graph

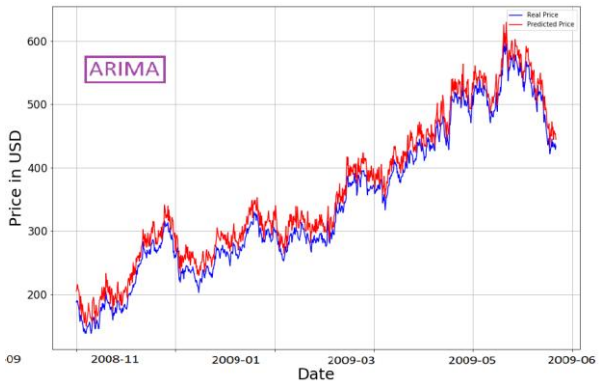

Figure 10: ARIMA Line Graph

Where Axis-X shows the date and axis-Y shows the stock's price at that date.The graphs of LSTM and IndRNN observations which show the difference between the forecasted data and the real data between the years 2005 and 2009 are represented in the following figures:

Figure 11: LSTM Prediction Line Graph

Figure 12: IndRNN Prediction Line Graph

D. Fine-tuning IndRNN

By replacing the activation function used in IndRNN with an advanced activation function called PReLU, the accuracy will be increased as it doesn't saturate when approaching the slope. We replaced the activation function of IndRNN and tested this suggested upgraded model on the same dataset. We also used the same number of epochs we used with LSTM and IndRNN (200). We got a better accuracy than IndRNN and the other approaches we tested.

The hyperparameters used in our fine-tuned IndRNN were as follows:

Table 5: Hyperparameters of our proposed modified IndRNN network

Hyperparameter	Value
Deep Learning Model	Fine-Tuned IndRNN
Timesteps	40
Optimizer	Adam
Learning rate	0.001
Batch size	40
Epochs	200
Activation function	PReLU
Number of the hidden layers	5
Neurons in the hidden layer	40

The MSE of our suggested updated fine-tuned IndRNN method, when tested on the same Tata Motors Ltd. dataset, was 44.16 which is less than the MSE of IndRNN.

The graph of our upgraded fine-tuned IndRNN observation which shows the difference between the forecasted data and the real data between the years 2006 and 2009 is as follows:

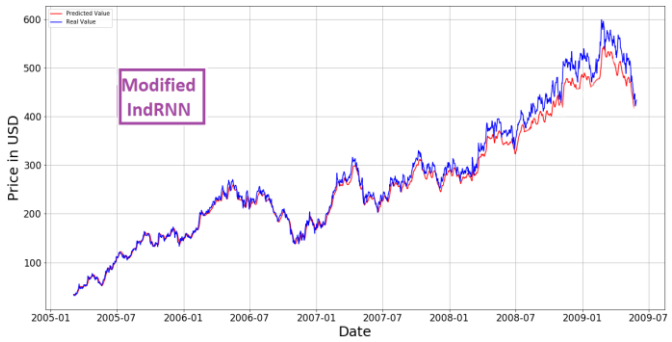

Figure 13: Fine-Tuned IndRNN Prediction Line Graph

V. VERIFYING OUR CLAIM

To verify our results and verify the accuracy of the tested algorithms, we had to test the same algorithms and to follow the same steps but with a different dataset. We downloaded a dataset contains the stock prices of the Amazon.com company from 1996 to 2019 with 5530 observations.[1] Here we have performed the analysis using the same five algorithms which are MA, ARIMA, LSTM, IndRNN, and our updated proposed IndRNN.

The mean squared error obtained for each of the models for this Amazon.com of the multinational sector is given in the table below.

Algorithm	MSE
MA	137
ARIMA	79
LSTM	75
IndRNN	69
Fine-tuned IndRNN	57

LSTM performed better compared to the other two linear models (MA and ARIMA). IndRNN performed better than the three methods. However, by fine-tuning it by changing its activation function, we got lower MSE which means better accuracy. This verifies that our suggested method outperforms the currently used methods in predicting time-series data.

VI. CONCLUSION

Deep learning is becoming a useful tool for forecasting statistic information like stock market prices etc. In our work, we have used two linear functions (MA and ARIMA) and three Deep Learning architectures (LSTM, IndRNN, and our Fine-Tuned IndRNN) for the stock price prediction of the NSE, Auto Mobile sector. We trained the neural networks with the stock prices of Tata Motors Ltd. [1]. From the results obtained, it's concluded that DL architectures are giving better results in comparison with MA and ARIMA models. Our work hasn't explored the gain of the usage of a hybrid network that links more than one network to make a new model for prediction. Our work can be further extended by capturing irregular behaviors and changes in the stock market by using techniques like CNN, particularly that the historical data isn't the only factor affecting the price but also unpredictable factors such as political events, etc. We can extract structured events from financial news and utilize external data to obtain event embedding. Then, combine event embedding and price values to forecast the stock trend.

REFERENCES

1. Amazon.com, inc. (amzn) stock historical prices & data – yahoo finance. https://in.finance.yahoo.com/quote/AMZN/history?p=AMZN%20on%2002/19/2020)
2. Tata motors ltd. (tatamotors.ns) stock historical prices & data – yahoo finance. https://in.finance.yahoo.com/quote/TATAMOTORS.NS/history?p=TATAMOTORS.NS
3. Albano, M., Jantunen, E., and Papa, G. The MANTIS Book: Cyber-Physical System Based Proactive Collaborative Maintenance. River Publishers Series in Automation, Control and Robotics. River Publishers, 2019.
4. Ding, B., Qian, H., and Zhou, J. Activation functions and their characteristics in deep neural networks. In 2018 Chinese Control And Decision Conference (CCDC) (2018), IEEE, pp. 1836–1841.
5. Dutta, S. Reinforcement Learning with TensorFlow: A beginner's guide to designing self-learning systems with TensorFlow and OpenAI Gym. Packt Publishing Ltd, 2018.
6. Hamilton, J. D. Time series analysis, vol. 2. Princeton New Jersey, 1994.
7. Hiransha, M., Gopalakrishnan, E. A., Menon, V. K., and Soman, K. Nse stock market prediction using deep-learning models. Procedia computer science 132 (2018), 1351–1362.
8. Li, S., Li, W., Cook, C., Zhu, C., and Gao, Y. Independently recurrent neural network (indrnn): Building a longer and deeper rnn. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018).
9. Nicolas, P. Scala for Machine Learning. Community experience distilled. Packt Publishing, 2015.
10. P, N. C., Gopal, D. V. K., R, V., Nambiar, L., and Menon, D. G. Predicting market prices using deep learning techniques. International Journal of Pure and Applied Mathematics 118, 20 (2018), 217–223.
11. Pankratz, A. Forecasting with Univariate Box - Jenkins Models: Concepts and Cases. Wiley Series in Probability and Statistics. Wiley, 2009.
12. Parmar, I., Agarwal, N., Saxena, S., Arora, R., Gupta, S., Dhiman, H., and Chouhan, L. Stock market prediction using machine learning. In 2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC) (2018), IEEE, pp. 574–576.
13. Wang, S.-H., Muhammad, K., Hong, J., Sangaiah, A. K., and Zhang, Y.-D. Alcoholism identification via convolutional neural network based on parametric relu, dropout, and batch normalization. Neural Computing and Applications 32, 3 (2020), 665–680.
14. Wu, W., Han, F., Song, G., and Wang, Z. Music genre classification using independent recurrent neural network. In 2018 Chinese Automation Congress (CAC) (2018), IEEE, pp. 192–195.
15. Xu, B., Wang, N., Chen, T., and Li, M. Empirical evaluation of rectified activations in convolutional network. arXiv preprint arXiv:1505.00853 (2015).

AUTHORS PROFILE

Mr. Ahmad Alsharef received his Bachelor of Engineering in Information Technology with a specialization in Computer Networks and Operating Systems from Damascus University, Syria. And Also he is going to complete his M.Tech with a specialization in Data Analytics from Kalinga Institute of Industrial technology deemed to be university, Bhubaneswar, Odisha, India. He had 2 years of experience in the software industry. He served as a volunteer lecturer at Kalinga Institute of Social Sciences, Odisha, India. His areas of research interest are Networks Security, Software Engineering, Data Mining, Deep Learning, Data Analytics, Big Data, Machine Learning, etc.

Dr.Prachet Bhuyan received his B.E in Computer Science & Engineering from Utkal University, Bhubaneswar, Odisha, India. M.Tech with specialization in Computer Science & Engineering from VTU, Belgaum, Karnataka, India. Ph.D in Computer Science & Engineering from KIIT Deemed to be University, Bhubaneswar, Odisha, India.

He served in various capacities from Lecturer to Head of the department in Vemana Institute of Technology, Bangalore over a decade. Also, he has served abroad at Caledonian College of Engineering (An affiliation of Glasgow Caledonian University), Muscat, Sultanate of Oman in teaching and research. He joined KIIT Deemed to be University from September 2007 and currently working. He is also the active member of Industry Engagement (IE) of KIIT Deemed to be University and was a member of the Government recognized KIIT-TBI as an executive committee member of TDB. He has various significant publications to his credit in leading Journals indexed in Scopus and International conferences. He has co-authored a Book titled "BigData-A Primer", published by Springer. His areas of research interest are Software Engineering, SOA, Software Testing, Soft computing, Grid Computing, Data Analytics, Cloud Computing, Machine Learning, and IoT. He has been a member of various technical organizations like IEEE, IET, ISTE, ISC and he is a member of ISAET (International Scientific Academy of Engineering & Technology) in its Scientific Technical Committee and Editorial Board on Engineering and Technology. He has successfully guided many industry-based students' projects, in collaboration with Microsoft, SAP Lab, Novell, IBM, Infosys, HighRadius Preva Systems, Unisys, Aakar i3, and Cognizant. Guided 11 M.Tech theses, many B.Tech projects and also Ph.D. thesis. He is active in consultancies and project guidance on various domains using IoT, AI, Analytics such as projects in smart city, smart home, green building, smart greenhouse, smart municipality, smart health care, smart vehicle pollution control to name a few. And has many recognition to his credit from the corporate world words under University Academic Alliances program.

Dr. Abhishek Ray received his B.E (CS&E) from Utkal University, Odisha. M.Tech with specialization in CS from REC, Rourkela, Odisha. Completed his Ph. D. in Computer Science & Engineering from KIIT University, Bhubaneswar, Odisha. He had more than 20 years of vast experience in teaching at UG and PG level. He served as a faculty of the Department of Computer Science and Engineering at Gandhi Institute of Engineering & Technology, Gunupur, Odisha, from 1998 to 2005. He joined KIIT University from July 2005 and has risen from Sr. Lecturer to Associate Professor. He is also a member of Industry Engagement Cell (IEC) of KIIT University. He has various publications to his credit in leading Journals indexed in Scopus listing and International conferences. His areas of research interest are Program Slicing, Software Testing, Soft computing, Bigdata, Cloud Computing, IoT, etc. He is a member of various technical organizations like IET, ISTE, ISC and he is a member of ISAET (International Scientific Academy of Engineering & Technology) in its Scientific Technical Committee.