2020 IEEE REGION 10 CONFERENCE (TENCON)
Osaka, Japan, November 16-19, 2020

Indian Stock Market Prediction using Deep Learning

Ayan Maiti
Department of Mathematical and Computational Sciences
National Institute of Technology Karnataka
Surathkal, India
ayan.182ma008@nitk.edu.in

Pushparaj Shetty D
Department of Mathematical and Computational Sciences
National Institute of Technology Karnataka
Surathkal, India
prajshetty@nitk.edu.in

***Abstract*—In this paper, we predict the stock prices of five companies listed on India's National Stock Exchange (NSE) using two models- the Long Short Term Memory (LSTM) model and the Generative Adversarial Network (GAN) model with LSTM as the generator and a simple dense neural network as the discriminant. Both models take the online published historical stock-price data as input and produce the prediction of the closing price for the next trading day. To emulate the thought process of a real trader, our implementation applies the technique of rolling segmentation for the partition of training and testing dataset to examine the effect of different interval partitions on the prediction performance.**

***Index Terms*—LSTM, GAN, generator, discriminant, neural networks, technical indicators, rolling segmentation**

I. Introduction

The prediction of stock prices and the direction of the price movement is a widely studied topic in many fields including trading, finance, statistics, and computer science. Typically, the professional stock market traders use fundamental and/or technical analysis of stocks to analyze the market elements/indices to make investment decisions [1]. Fundamental analysis is the conventional method involving the study of company fundamentals such as sales, revenue and expenses, net profit, debts, annual growth rates, and so on, whereas technical analysis, is exclusively based on the study of historical price movements. Practitioners of technical analysis refer to the price charts to study price patterns and use price data in different calculations to forecast future price movements. The technical analysis paradigm assumes, there exists an inherent correlation between the performance of the company and its stock price and therefore, it is used to determine the timing of buying and selling the shares of the company.

Time-series prediction problems like the prediction of stock prices are a difficult type of predictive modeling problem. Such problems usually involve the presence of sequence dependence among the input variables [2]. A robust type of neural network equipped to handle the sequence dependence is the Recurrent Neural Networks (RNNs). The Long Short-Term Memory network or LSTM network is a type of RNN and is used extensively in time-series problems because of its capability to handle dependency between both long and short distance sequences [3]. Generative adversarial network (GAN) has been successfully applied to a wide range of fields, such as image in painting, semantic segmentation, and video prediction. GAN consists of a discriminative network D, which learns to distinguish if a given data instance is real or not, and a generative network G, which learns to confuse D by generating high-quality data [4]. This approach has also been used for the prediction of the high-frequency stock prices of the Shanghai and Shenzhen stock exchanges of the Chinese stock market [5]. However, it has not been used to predict the daily stock prices of the Indian stock market as per our knowledge.

Our contributions in this paper are as follows: 1. Implemented two models for daily frequency stock price prediction- LSTM and GAN 2. Performed rolling segmentation on the training set and testing set of the raw data to investigate the effect of the model parameter update cycle on stock forecast performance.

We found that LSTM outperformed GAN significantly with an error of 0.074 compared to 0.32 obtained by GAN. It implies that the price predicted by LSTM differs from the original price by 7.4% whereas for GAN it is 32%.

II. Related Works & Background

In the recent times, deep learning is emerging as a promising area of research and has been applied for classification and prediction tasks in various domains like text classification, image processing, speech recognition, and many others. It has also been applied in the prediction of time-series since it is suitable to address chaotic data, randomness, and non-linearity [1]. For example, Chong et al. [6] have utilized the capability of deep neural networks to extract abstract features from data by using a deep feature learning-based stock market prediction model, which extract information from the stock return time series without relying on prior knowledge of the predictors and tested it on high-frequency data from the Korean stock market.

Lately, LSTM network, which is suitable for learning temporal patterns, is extensively utilized for various tasks of time-series analyses [7]. LSTM is preferred over the conventional RNN as it overcomes the problem of vanishing (or exploding) gradients and as it can effectively learn long-term dependencies through memory cells and gates. Thus, many studies on financial time-series modeling are conducted using LSTMs.

However, Zhou et al [5] have adopted GAN to predict stock prices on highly volatile stocks in the China Stock Exchange market. GAN has not been applied much to financial markets across the globe and certainly not on the Indian Stock Market to the best of our knowledge. In this work, we implement both

978-1-7281-8455-5/20/$31.00 ©2020 IEEE

LSTM and GAN models to predict the stock prices on the Indian market and compare their performance.

A. LSTM

LSTM can be trained to learn dependencies ranging over very long time intervals of time. It beats the vanishing gradients problem faced by a general RNN by substituting the ordinary neuron by a complex architecture called the LSTM unit or block [3]. The working of LSTM can be summarised by the following set of equations.

$$\begin{aligned} z_t &= \tanh(W^z x_t + R^z h_{t-1} + b^z) \\ i_t &= \sigma(W^i x_t + R^i h_{t-1} + b^i) \\ f_t &= \sigma(W^f x_t + R^f h_{t-1} + b^f) \\ o_t &= \sigma(W^o x_t + R^o h_{t-1} + b^o) \\ s_t &= z_t \cdot i_t + s_{t-1} \cdot f_t \\ h_t &= \tanh(s_t) \cdot o_t \end{aligned} \tag{1}$$

where i_t denotes the input gate and o_t denotes the output gate. The forget gate, memory cell, and hidden state are denoted by f_t, s_t, and h_t, respectively. The σ and $\tanh$ functions are defined in 2 and 3 respectively.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2}$$

$$\tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{3}$$

B. GAN

Generative adversarial networks was introduced by Goodfellow et al. [4], where image patches are generated from random noise using two networks trained simultaneously. It has shown very good results when applied to various tasks particularly image synthesizing. GAN belong to the family of generative models. GAN models learn the given set of distribution (training data) and produce data similar to the original data distribution. The model can either generate samples from a distribution or represent an actual distribution. Traditionally, training models require to have a dataset with labels for all the data. However, generative models can be trained with a dataset holding a mixture of both unlabeled and labeled data, thus eliminating the constraint caused by the unavailability of information. The generative adversarial network comprises of two models, a Generator and a Discriminator. The purpose of the generator is to take noise as input, which is usually a uniform distribution, and generate samples that look like they were drawn from a distribution representing our target distribution. Samples are then drawn from a distribution representing our target distribution (real data), and the samples produced by our Generator (fake data) are fed into the Discriminator which then predicts which ones are real and which ones are fake. The output produced by the discriminator is then propagated to the generator to update, improve, and generate more realistic samples. For an overview of the network see Fig. 1.

1) Generator: We can define the Generator function as a function G that is differentiable with respect to its parameters θ_G and input data z. z is considered as the noise drawn from a distribution P_z, e.g. Gaussian distribution. The generator will then map z to samples $G(z)$ that form the generator's distribution P_g which can be expected to be the same as the true data distribution P_{data} after training. The mapped samples $G(z)$ are then put into the discriminator and the generator is then trained to deceive the discriminator by making it give high probabilities to the generated samples.

2) Discriminator: Similarly, we can define the discriminator function D differentiable with respect to its parameters θ_D and input x and $G(z)$, where x is real data samples drawn from the P_{data} distribution. The discriminator then produces the probability of the sample belonging to the real data distribution P_{data}. The discriminator will then be trained continuously, in a supervised manner, so that it gives high probability to real data samples and a low probability to the samples generated by the Generator.

3) Cost Function: Mathematically, the generator and discriminator cost functions are related by 4.

$$\min_G \max_D (\mathbb{E}_{X \sim P_{data}(x)}[logD(x)] + \mathbb{E}_{Z \sim P_{data}(z)}[log(1 - D(G(z)))]) \tag{4}$$

where $\mathbb{E}_{X \sim P_{data}(x)}$ and $\mathbb{E}_{Z \sim P_{data}(z)}$ are the expected values of the P_{data} and P_z distributions.

We can apply Stochastic Gradient Descent (SGD) to solve this optimization problem (see 5). The optimization is difficult and the space of solutions (sets of weights) should contain global optima. On other hand a local optima is relatively easier to find but does not perform well.

$$W_{t+1} = W_t - \eta \frac{\partial L}{\partial W} \tag{5}$$

where W is the set of weights and W_{t+1} and W_t are set of weights at iterations $t + 1$ and t respectively. L is the cost function, η is the learning rate.

The amount by which the set of weights in the model change to reach the space of solutions, or the step size, is called the learning rate. It is the most important hyperparameter to tune to achieve good performance on the problem.

In this tutorial, you will discover the learning

III. Methodology

A. Dataset

We obtained historical data of 5 stocks from Yahoo Finance (https://in.finance.yahoo.com/) that are being actively traded on India's National Stock Exchange (NSE). To maximize the available data-points, the stocks were selected such that their historical data of range from January 1, 1996 to May 15, 2020. There are 253 trading days in a year for the NSE, so on an average we get 5500 data-points for every stock at an interval of 1 day. The historical price data contains 5 features- 'Open Price', 'Maximum Price', 'Minimum Price',

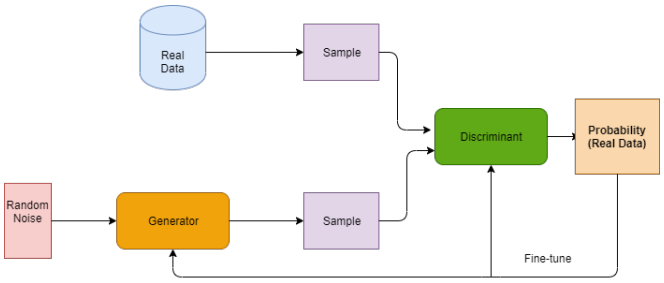

Fig. 1. A Generative Adversarial Network

'Closing Price' and 'Trading Volume'. Also, we selected 9 additional technical indicators as feature subsets after a rigorous literature review. Many investors in the stock market study these technical indicators derived from the past stock-price data and use it as the signal for future market trends [1] [5]. So, the input data **X** at each day (say, X_T) consists the total of 13 basic indicators as listed in Table I.

TABLE I
BASIC INDICATORS FOR PREDICTION

Technical Indicators
Opening Price
Maximum Price
Minimum Price
Trading Volume
Moving Averages
Bollinger Bands
Exponential Moving Averages
PPSR
On Balance Volume
Price Rate of Change
Average Directional Movement Index
Relative Strength Index
MACD

B. Data Partition

Traditionally, data is split for training and testing the models using fixed partitioning method. Nevertheless, the trading pattern of the stock market changes regularly, for example, investors at times favour stocks with high volatility whereas sometimes choose to invest in technology stocks. Accordingly, we should update the model parameters frequently to adjust to the change of market pattern. For making our experiments closer to real transactions, we carry out rolling segmentation on the training set and testing set of the experimental data. This technique of data partitioning was used in [5]. As Figure 2 shows, at the start, we select the first M days as the training set and the next N days as the testing set. For the next round, we roll the time window ahead for N days, that is, choosing the $(N+1)^{th}$ day to the $(M+N)^{th}$ day as the training set and the $(M+N+1)^{th}$ day to the $(M+2N)^{th}$ day as the testing set. We repeat this until all the data has been experimented. In other words, this N can be regarded as the model update cycle, and M is the size of the corresponding training data.

C. Lookback Days

Let X_i denote the set of basic technical indicators and Y_i represent the closing price of one stock for a 1-day interval at time i $(i = 1, 2, ..., T)$, where T is the maximum lag of time (lookback days). Given the historical basic indicators data X $(X = X_1, X_2, ..., X_T)$ and the past closing price Y $(Y = Y_1, Y_2, ..., Y_T)$, we try to predict the closing price Y_{T+1} for the next 1-day time interval. For this project, we have chosen T as 90 and 210 for the LSTM model and 45 for both LSTM and GAN models based on computational time and limitations. We have tested the LSTM model with T=210 for smaller values of M and N (refer section III-B) and with T=90 for all values of M and N. GAN, model was tested only on T=45 for all values of M and N.

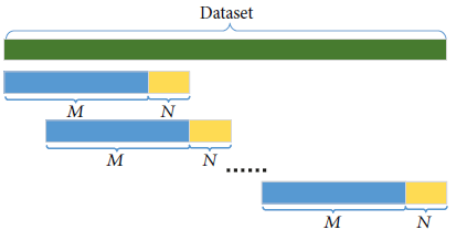

Fig. 2. Rolling segmentation on dataset. The green bar shows the entire dataset, the blue bar shows the training set for one round experiment, and the yellow bar is the corresponding testing set. Image Source- [5]

D. LSTM Model Architecture

We have used a single layer of LSTM internal units followed by a Dropout layer. The activation function in LSTM is tanh and the loss function used is *Mean Squared Error*. The values for the hyper-parameters are mentioned in Table II.

E. GAN Model Architecture

As discussed earlier a GAN is a network of 2 models- generator G, and discriminator D. Here, we used the loss functions defined and used by Zhou et al. [5].

a) Generator: We have used a layer of LSTM followed by a dropout layer as our generator. To make the discriminative model D as "confused" as possible, so that D does not distinguish between actual and closing price, the generative model G should have an adversarial loss which should be minimized such that D will not discriminate the prediction correctly. Consequently, we label actual closing price $\bar{Y}$ as class 1 and predicted closing price $\hat{Y}$ as class 0, and define the adversarial loss for G as 6

$$L_{adv}^{G}(\hat{Y}) = L_{sce}(D(\hat{Y}), 1) \tag{6}$$

where, L_{sce} is the sigmoid cross-entropy loss, defined as 7

$$L_{sce}(A, B) = -\sum_{i} B_i \log(sigmoid(A_i)) + (1 - B_i) \log(1 - sigmoid(A_i)) \tag{7}$$

TABLE II
HYPER-PARAMETER VALUES FOR LSTM MODEL

Internal Units	Learning rate	Dropout Rate	Batch Size	Epochs
495	0.00385	0.35302	60	600

Nevertheless, minimizing adversarial loss solely cannot assure satisfying predictions. For example, G might produce samples to "confuse" D, without being similar to $\bar{Y}_{T+1}$, and then D will train to distinguish these samples, directing G to generate other "confusing" samples, and this futile process will continue. To fix this issue, the generative model G must decrease the prediction error loss; that is, L_p loss

$$L_P(\bar{Y}, \hat{Y}) =\mid \hat{Y} - \bar{Y} \mid \tag{8}$$

Besides, direction of stock price prediction is also critical for trading, so we define direction prediction loss function L_{dpl} as:

$$L_{dpl}(\bar{Y}, \hat{Y}) =\mid sgn(Y_{T+1}\hat{} - Y_T) - sgn(Y_{T+1} - Y_T) \mid \tag{9}$$

where sgn represents sign function. We combined all these losses defined, and achieve the final loss on G:

$$L_G(X, Y) = L^G_{adv}(\hat{Y}) + L_P(\bar{Y}, \hat{Y}) + L_{dpl}(\bar{Y}, \hat{Y}) \tag{10}$$

We perform one SGD step on G to minimize L_G(X, Y) while keeping the weights of D fixed. The values of the hyper-parameters for G are mentioned in Table III.

b) Discriminator: The Discriminator is a simple dense layer of leakyRelu activation functions followed by a sigmoid function layer. The only job of D is to determine if the input is $\bar{Y}$ (real) or $\hat{Y}$ (generated), our target loss here is equal to the adversarial loss for D. We keep the weights of G fixed, and perform one SGD iteration on D to reduce the target loss:

$$\begin{aligned} L_D(X, Y) &= L^D_{adv}(\bar{Y}, \hat{Y}) \\ &= L_{sce}(D(\hat{Y}), 0) + L_{sce}(D(\bar{Y}), 1) \end{aligned} \tag{11}$$

The values of the hyper-parameters for D are mentioned in Table IV.

F. Evaluation Metrics

Based on a specific model, for every stock at each time t, a prediction is obtained for the next time point $t+1$. Assuming the total number of time points to be tested is T_0; we employed the following measure to gauge the performance of the different models.

1) Root Mean Squared Relative Error (RMSRE):

$$RMSRE = \sqrt{\frac{1}{T_0}\sum_{t=1}^{T_0}\left(\frac{\hat{Y}_{t+1} - Y_{t+1}}{Y_{t+1}}\right)^2} \tag{12}$$

RMSRE is used as a yardstick for the predictive strength or the prediction consensus. A low RMSRE signals that the prediction concurs with the actual data. The reason why this work preferred RMSRE over Root Mean Squared Error (RMSE) is that RMSRE simplifies the uniform comparison of the results of 5 stocks.

2) Direction Prediction Accuracy (DPA):

$$DPA = \frac{1}{T_0}\sum_{t=1}^{T_0} I_t \tag{13}$$

where

$$I_t = \begin{cases} 1, & \text{if } (Y_{t+1} - Y_t)\left(\hat{Y}_{t+1} - Y_t\right) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

DPA calculates the per unit accuracy with respect to the series trend. We can find the percentage accuracy by multiplying DPA by 100. A high DPA assures more winning trades. We have used the same metrics that are used in [5] to measure the performance of the models.

IV. RESULTS

To examine the effect of the model update cycle on the predictive performance, we have tested our models on look-back of 45, 90, and 210 days.

A. 45 days Look-back

To test LSTM and GAN prediction models, we have taken $M \in \{400, 700, 1200\}$ for $N = 300$ and $M \in \{800, 1400, 2400\}$ for $N = 600$. Tables V and VI show the average values of RMSRE and DPA with different (M, N). The numbers clearly indicate that LSTM has performed better than GAN model in terms of RMSRE metric. Although we have used specific loss function component for direction in GAN there has been no significant improvement in DPA. There is strong possibility that the models have not achieved the optimum set of weights to solve the optimization problem and produce good results. The best RMSRE has been observed for $(M, N) = (700, 300)$, in LSTM model.

B. 90 days Look-back

We have also tested model LSTM on past 90 days data. For this purpose we have taken $M \in \{400, 700, 1200\}$ for $N = 300$ and $M \in \{800, 1400, 2400\}$ for $N = 600$. The result is shown in Tables VII and VIII. We can observe that the best RMSRE was again obtained for $M = 1200$ and $N = 300$. The model performance for DPA is very poor.

C. 210 days Look-back

We could test the model LSTM only on $M \in \{400, 700\}$ and N = 300 as 210 days look-back demands more sophisticated computational resources. The result is presented in Tables IX and X. The best result is obtained with $M = 700$ and $N = 300$ whereas there is no improvement for DPA.

TABLE III
HYPER-PARAMETER VALUES FOR GENERATOR

Internal Units	Learning rate	Dropout Rate	Batch Size	Epochs
28	0.00266	0.1077	200	3500

TABLE IV
HYPER-PARAMETER VALUES FOR DISCRIMINATOR

Internal Units	Learning rate	Batch Size	Epochs
504	0.002117	200	3500

TABLE V
SUMMARY OF RMSRE WITH DIFFERENT (M,N). THESE FIGURES ARE AVERAGE VALUE OF 5 STOCKS ON BASIS OF LAST 45 DAYS.

	N=300			N=600		
Model	M=400	M=700	M=1200	M=800	M=1400	M=2400
LSTM	0.099581	0.078011	0.083184	0.134927	0.140068	0.108776
GAN	0.333444	0.324703	0.425867	0.404987	0.469092	0.626753

TABLE VI
SUMMARY OF DPA WITH DIFFERENT (M,N). THESE FIGURES ARE AVERAGE VALUE OF 5 STOCKS ON BASIS OF LAST 45 DAYS.

	N=300			N=600		
Model	M=400	M=700	M=1200	M=800	M=1400	M=2400
LSTM	0.516034	0.506645	0.506106	0.506660	0.504132	0.503534
GAN	0.503628	0.508824	0.524844	0.516667	0.50981477064	0.526689

TABLE VII
SUMMARY OF RMSRE WITH DIFFERENT (M,N). THESE FIGURES ARE AVERAGE VALUE OF 5 STOCKS ON BASIS OF LAST 90 DAYS.

	N=300			N=600		
Model	M=400	M=700	M=1200	M=800	M=1400	M=2400
LSTM	0.109988	0.077702	0.073884	0.135652	0.128104	0.096968

TABLE VIII
SUMMARY OF DPA WITH DIFFERENT (M,N). THESE FIGURES ARE AVERAGE VALUE OF 5 STOCKS ON BASIS OF LAST 90 DAYS.

	N=300			N=600		
Model	M=400	M=700	M=1200	M=800	M=1400	M=2400
LSTM	0.505599	0.504585	0.505893	0.506125	0.504858	0.502165

TABLE IX
SUMMARY OF RMSRE WITH DIFFERENT (M,N). THESE FIGURES ARE AVERAGE VALUE OF 5 STOCKS ON BASIS OF LAST 210 DAYS.

	N=300 M=400	N=300 M=700
LSTM	0.108374	0.081476

TABLE X
SUMMARY OF DPA WITH DIFFERENT (M,N). THESE FIGURES ARE AVERAGE VALUE OF 5 STOCKS ON BASIS OF LAST 210 DAYS.

	N=300 M=400	N=300 M=700
LSTM	0.502477	0.500712

V. Conclusion

The results of the model update cycles on the predictive performance show that M=1200 and N=300 produce the best results of RMSRE=0.073884, obtained at look-back of 90 days. In other words, the predicted price had an average error difference of 7.3884% with respect to the actual price. However, the corresponding DPA was mere 0.505893, implying the model is not able to predict the direction of the price movement. We also found that increase in the number of look-back days did not have significant improvement of results on LSTM model. As we suspect that the models have not achieved optimum training, we present some ideas below to refine and continue our work forward:

1) The GAN model can be experimented on higher number of look-back days and experiment by changing the hyper-parameters to improve the performance.
2) Experiment by incorporating the loss function for direction accuracy used in GAN model into LSTM to check for any improvement in DPA. Also, new loss function can be designed to achieve better results.
3) We have used stock price data of daily frequency. Higher frequency data of every minute stock price can be obtained to train and test the performance of the models implemented in this project.

Acknowledgment

The authors would like to thank National Institute of Technology Karnataka, India, for the support during this work.

References

[1] I. K. Nti, A. F. Adekoya, and B. A. Weyori, "A systematic review of fundamental and technical analysis of stock market predictions," *Artificial Intelligence Review*, pp. 1–51, 2019.
[2] R. J. Frank, N. Davey, and S. P. Hunt, "Time series prediction and neural networks," *Journal of intelligent and robotic systems*, vol. 31, no. 1-3, pp. 91–103, 2001.
[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.
[5] X. Zhou, Z. Pan, G. Hu, S. Tang, and C. Zhao, "Stock market prediction on high-frequency data using generative adversarial nets," *Mathematical Problems in Engineering*, vol. 2018, 2018.
[6] E. Chong, C. Han, and F. C. Park, "Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies," *Expert Systems with Applications*, vol. 83, pp. 187–205, 2017.
[7] D. H. D. Nguyen, L. P. Tran, and V. Nguyen, "Predicting stock prices using dynamic lstm models," in *International Conference on Applied Informatics*. Springer, 2019, pp. 199–212.