Comparative analysis of Vanilla LSTM and Peephole LSTM for stock market price prediction

Atharva Joshi
Department of Computer Engineering
College of Engineering, Pune
Pune, India
atharvamj18.comp@coep.ac.in

Dr. Pradeep K. Deshmukh
Associate Professor
Department of Computer and IT
College of Engineering, Pune
Pune, India
pkd.comp@coep.ac.in

Jay Lohokare
Department of Computer Science
State University of New York, Stony Brook
Stony Brook, USA
jlohokare@cs.stonybrook.edu

***Abstract*—Stock market price prediction is one of the most challenging problems in the field of time series forecasting because of the chaotic and volatile nature of stock market data. The currently existing technology is evolving in terms of algorithmic sophistication to provide better ways to predict behavior in highly volatile and noisy environments. A new trend in the machine learning and pattern recognition fields suggests that time-series prediction should be done using a deep nonlinear topology. Many papers on stock market price prediction have analyzed various machine learning algorithms like Artificial Neural Networks, ARIMA, CNN, RNN, SVM classifier, Fuzzy classifier, Random forest, and Bayesian model. Most suggest that a variant of RNN called LSTM outperforms the other algorithms. However, there is still a lesser-explored area within LSTM variants that can further be explored.**

This paper compares the performance of Vanilla LSTM with the other LSTM variant called Peephole LSTM. The experiments revealed that Peephole LSTM outperforms Vanilla LSTM by a significant margin.

***Index Terms*—Stock market, time series, Vanilla LSTM, Peephole LSTM**

I. Introduction

Attempting to forecast the future price of business stock or other asset class traded on a securities exchange is known as stock market prediction. A good forecast of a stock's future price might result in a significant profit. According to the efficient-market theory, stock prices represent all existing accessible information, as well as any price fluctuations that are not dependent on recently disclosed information are therefore fundamentally unforeseeable.

Stock market forecasting has progressed into the technical arena with the invention of the digital computer. The most often used algorithms for predicting stock prices and trends are Artificial Neural Networks (ANN) and Support Vector Machines (SVM). ANNs may be conceived of as approximators for mathematical functions. The feed-forward network, which uses the backpropagation of errors strategy to modify the network parameters, is the most frequent type of ANN used for stock market prediction. Another kind of ANN that is superior at stock price prediction is the Long Short Memory Network, a variant of the Recurrent Neural Network.

978-1-6654-6883-1/22/$31.00 ©2022 IEEE

Despite the variety of algorithms, ranging from linear regression to advanced deep learning algorithms, stock price prediction remains an open problem.

The performance of two LSTM variants (Vanilla LSTM and Peephole LSTM) is compared in this study.

II. Related Works

This research focuses on the price prediction aspect concerning stock market. In the last decade, significant research has been done in this area.

A model was created to function as a rolling window by D. M. Q. Nelson et al. in [1] . A new neural network was built at the end of each trading day i.e. a new set of weights were defined using a new set of training and validation data. The LSTM-based model was shown to have lower risks than the other choices while looking at maximum losses. The study doesn't account for the complexities of financial markets, such as transaction cost, execution booking and timing.

When predicting time series data, the research done by S. Siami-Namin et al. in [2] examined the accuracy of ARIMA and LSTM as representative methodologies. Compared to ARIMA, the LSTM-based algorithm improved prediction by 85 percent on average. However, the model in this study needs to be tuned carefully because a slight change in the parameters like epochs and batch size may worsen the performance.

For determining the direction of the Nifty-500 indices, S. A. Dwivedi et al. compared time series forecasting models like SARIMA, LSTM and CNN in [3]. Machine-learning-based prediction models were shown to perform considerably worse than LSTM-based deep learning models.

Attention-based multi-input LSTM model was studied by Li et al. in [4]. Machine-learning-based prediction models were demonstrated to be much inferior than LSTM-based deep learning models in terms of performance. The attention factors in the proposed model has a significant impact on the prediction. They need to be treated properly so as to get better predictions.

LSTM networks were used to predict out-of-sample directional motions by Fischer et al. in [5]. The potential of an LSTM network to extract significant information from noisy financial time series data was demonstrated in this research.

A hybrid modeling approach for stock price prediction involving building several models based on ML and DL was proposed by Mehtab et al. in [6]. The study showed that the models based on DL are far more capable than their ML counterparts in learning and extracting the properties of time series data.

An LSTM-based classification algorithm was used by Liu et al. in [7] to forecast time series trends. The newly proposed model is efficient compared with the conventional ARX model, which is still common today. The results outperform typical autoregressive models and even arbitrarily predicted outcomes on market datasets, which are akin to random walk sequences. The stock price data used in this study is challenging to deal with in time series research since it resembles a random walk sequence.

A new LSTM-based neural network model for estimating a company's stock price was proposed by Rokhsatyazdi et al. in [8]. The proposed model uses one of the most robust algorithms, Differential Evolution (DE). The improved LSTM model outperforms the statistical approaches like SARIMA, ETS and NAIVE. The model is limited by cost with respect to mixed-type variables and computation time.

III. LSTM Variants

This research work compares the performances of two variants of Long Short Term Memory Networks (LSTMs): Vanilla LSTM (the normal LSTM) and Peephole LSTM (the LSTM with peephole connections).

A. Vanilla LSTM

The LSTM is an RNN that can learn long-term dependencies. It is specifically developed to avoid the problem of vanishing or exploding gradients. Time-series data is well-suited to an LSTM for classification and/or prediction. LSTM units come in various topologies, with a typical design that includes an input gate, a forget gate, a memory cell and an output gate. In addition, each LSTM cell calculates new hidden state and cell state values. The LSTM cell's mathematical formulation is shown below. [9]

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \tag{1}$$

$$i_t = \sigma_g(W_i x_t + U_f h_{t-1} + b_i) \tag{2}$$

$$o_t = \sigma_g(W_f x_t + U_o h_{t-1} + b_o) \tag{3}$$

$$\hat{c_t} = \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \tag{4}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \hat{c_t} \tag{5}$$

$$h_t = o_t \circ \sigma_h(c_t) \tag{6}$$

Here, x_t is an input vector to LSTM unit, i_t, f_t, o_t and $\hat{c_t}$ are the activation vectors of input gate, forget gate, output gate and cell unit respectively. c_t is a cell state vector. W and U are weight matrices. b is bias vector parameter. σ_g is sigmoid tangent function and σ_c, σ_h are hyperbolic tangent functions. The functions of LSTM gates and states are:

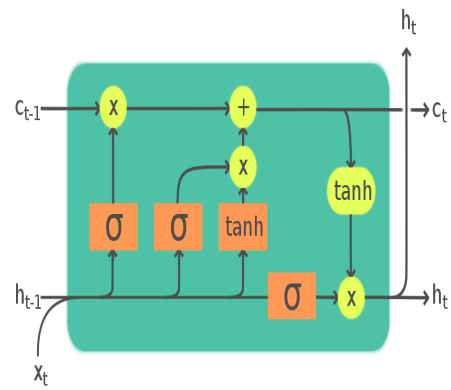

Fig. 1. LSTM Cell

- The cell state stores the model's long term memory
- The hidden state stores the working memory or the short term memory of the model.
- The input gate determines what needs to be saved in the model's long term memory using recent input and previous short term memory.
- The forget gate determines whether or not the model should retain long-term memory information. The long-term memory is multiplied by a forget vector to achieve this.
- The output gate produces the new short term memory by using the prior short term memory, the current input and the recently calculated long term memory.

B. Peephole LSTM

LSTM augmented by "peephole connections" from its internal cells to its multiplicative gates can learn the fine distinction between sequences of spikes spaced either 50 or 49 time steps apart without the help of any short training exemplars. Without external resets or teacher forcing, this LSTM variant also learns to generate stable streams of precisely timed spikes and other highly nonlinear periodic patterns. The mathematical formulation of Peephole LSTM is given below: [10]

$$f_t = \sigma_g(W_f x_t + U_f c_{t-1} + b_f) \tag{7}$$

$$i_t = \sigma_g(W_i x_t + U_i c_{t-1} + b_i) \tag{8}$$

$$o_t = \sigma_g(W_o x_t + U_o c_{t-1} + b_o) \tag{9}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + b_c) \tag{10}$$

$$h_t = o_t \circ \sigma_h(c_t) \tag{11}$$

Most of the equations are same as Vanilla LSTM with the difference being:

- c_{t-1} is used in place of h_{t-1} at most of the places

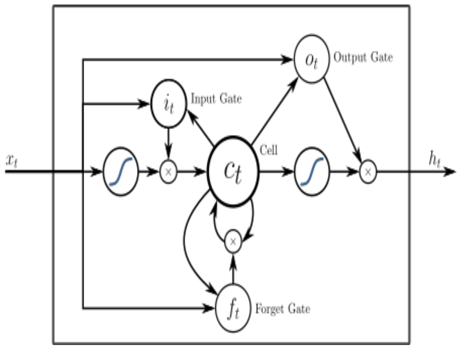

Fig. 2. Peephole LSTM Cell

- The arrows to output gate, forget gate and input gate from c (the memory cell) indicate the peephole connections.

IV. Methodology

A. Datasets

This paper analyses daily stock prices of five Indian stocks, namely MRF, BAJFINANCE, INFY, SAIL, and BHEL. The reason behind choosing these stocks is their price range and the dataset size. MRF and BAJFINANCE have a wider range of prices, whereas BHEL and SAIL have a narrower range of prices, and the range of INFY falls in between these price ranges. This is particularly useful to check how the algorithm performs in different price ranges. The details of stock ranges and dataset size are given in table I.

TABLE I
Dataset Range and Size

Stock	Min price (Rs.)	Max price (Rs.)	Dataset size
MRF	814.85	96973.85	4904
BAJFINANCE	4.06	7929.3	4904
INFY	0.76	1939.5	6602
BHEL	5.13	382.69	6599
SAIL	4	287.75	6602

B. Data Pre-processing

The historical data used in this work has been taken from the Yahoo Finance website. This data undergoes data pre-processing in order to train and test the algorithms. The dataset contains features like Open Price, Close Price, High Price, Low Price, Adj Close Price, and Volume. It was observed that keeping other prices to predict the closing prices has an overall negative impact on the predictions. This research deals with predicting the Close Price, so all the features except the Close Price were dropped. The data is scaled in (0, 1) using MinMaxScaler. MinMaxScaler outperforms other scalers like StandardScaler and RobustScaler. The data is then converted into vector form by rolling window fashion using a lookback period ranging from 10 to 60 with a difference of 10. The lookback period of k means we use previous k values to predict the next value.

C. Model Specifications

The model configuration used in this study is identical for both variants. The model configuration consists of three layers. The first layer is an input layer that specifies the input. The second layer is layer-specific to the variant, having 512 hidden units that take input from the first layer. The third (last) layer is a dense layer that takes input from the second layer and outputs the predicted value. It was observed that increasing the number of layers has an overall negative impact on the predictions. Whereas increasing the hidden units has an overall positive impact till hidden units reach 512. Thus, both the models have been trained with 512 hidden units.

The pseudocode for Vanilla LSTM can be given as follows:

```
def vanillaLSTM():
  model = Sequential()
  model.add(LSTM(units=512, ...))
  model.add(Dense(1))
  model.compile(...)
  return model
```

The pseudocode for Peephole LSTM can be given as follows:

```
def peepholeLSTM():
  model = Sequential()
  model.add(PeepholeLSTM(units=512, ...))
  model.add(Dense(1))
  model.compile(...)
  return model
```

Fig. 3 shows the configuration of the model for the look back period of 10.

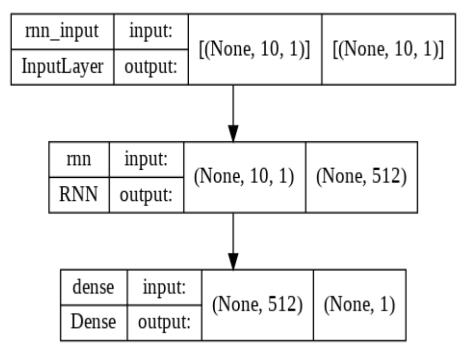

Fig. 3. Model Configuration

D. Training and Testing

The train and test split ratio of the dataset is 9:1. The data was trained on the configuration mentioned above with 50 epochs and a batch size of 32. The optimizer used was adam and the loss function used was mean squared error.

V. Results

The results obtained are compared in terms of errors. The metrics used to compare the errors of two algorithms are Root Mean Squared Error (RMSE) and Mean Absolute Error. The time step, also known as the lookback period, varied from 10 to 60. Observations of test RMSE and test MAE for each of the mentioned stocks i.e., MRF, BAJFINANCE, INFY, BHEL and SAIL are plotted in the figure 4, 5, 6, 7 and 8 respectively.

Table II displays the mean of test root mean squared error (RMSE). The average improvement in RMSE by Peephole LSTM over Vanilla LSTM is 6.04%

Table III displays the mean of test mean absolute error (MAE). The average improvement in MAE by Peephole LSTM over Vanilla LSTM is 9.85%.

Figures 9, 10, 11, 12 and 13 show the prediction plots for the stocks MRF, BAJFINANCE, INFY, BHEL and SAIL respectively. These plots are plotted for 200 unseen consecutive data points for each particular time series. The green colored curve represents the actual price. The blue and red colored curves represent the predicted prices using Peephole LSTM and Vanilla LSTM respectively.

TABLE II
AVERAGE TEST RMSE COMPARISON

Stock	Average Test RMSE		Improvement
	Vanilla LSTM	Peephole LSTM	
MRF	1741.06	1619.34	6.99%
BAJFINANCE	170.24	157.95	7.22 %
INFY	25.86	24.31	5.99%
BHEL	1.81	1.73	4.42%
SAIL	2.7	2.55	5.56%

TABLE III
AVERAGE TEST MAE COMPARISON

Stock	Average Test MAE		Improvement
	Vanilla LSTM	Peephole LSTM	
MRF	1493.91	1370.59	8.25%
BAJFINANCE	141.98	123.28	13.17%
INFY	19.55	17.77	9.10%
BHEL	1.36	1.26	7.35%
SAIL	1.93	1.71	11.40%

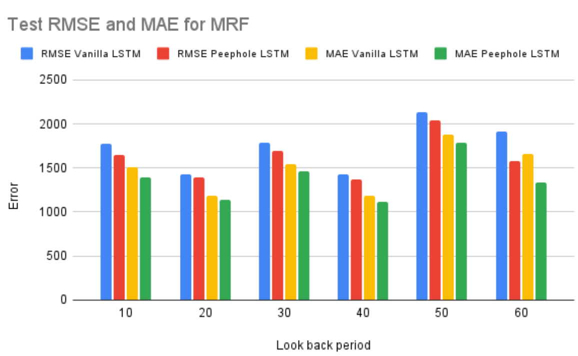

Fig. 4. Test errors for MRF

VI. DISCUSSION

As can be seen from the graphs plotted in figures 4, 5, 6, 7 and 8, most of the times the test RMSE generated by Peephole LSTM is lesser than the Vanilla LSTM. Only in 4 out of 25 cases does the Vanilla LSTM performs better than Peephole LSTM, with a very low margin. The test MAE also shows a similar trend. Moreover, the Peephole LSTM gives better results when the lookback period is between 40 and 60. The average improvement by Peephole LSTM over Vanilla LSTM in the case of RMSE is 6.04%, whereas it is 9.85% in the case of MAE. This clearly indicates the superiority of Peephole LSTM over Vanilla LSTM. The possible reason behind this is the ability of Peephole LSTM to identify highly nonlinear and periodic patterns in the time series.

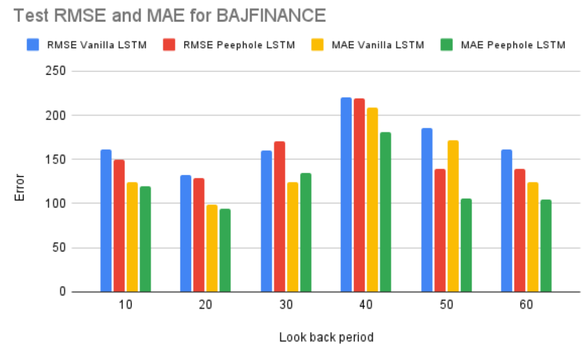

Fig. 5. Test errors for BAJFINANCE

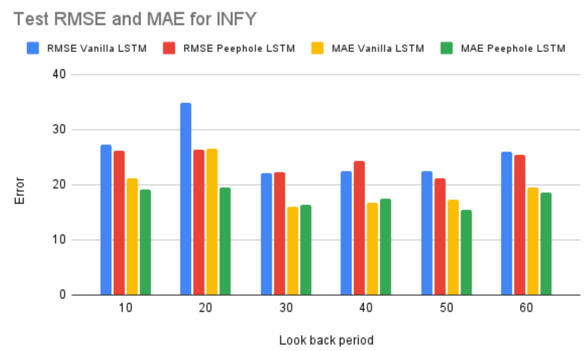

Fig. 6. Test errors for INFY

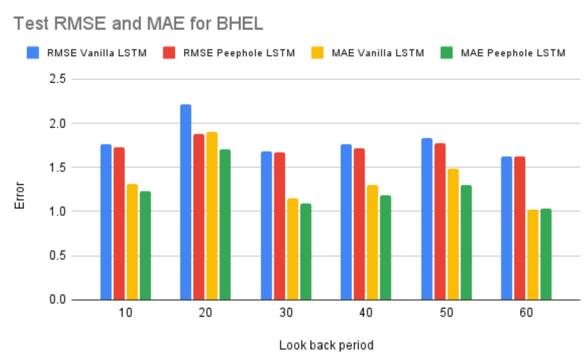

Fig. 7. Test errors for BHEL

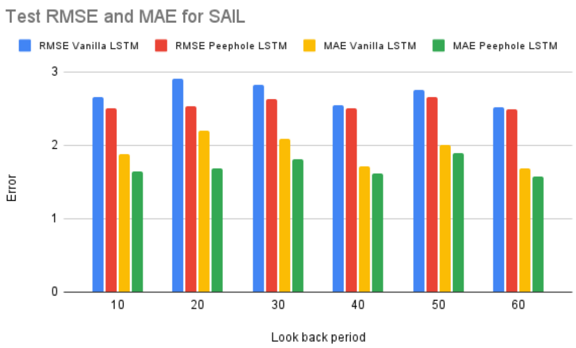

Fig. 8. Test errors for SAIL

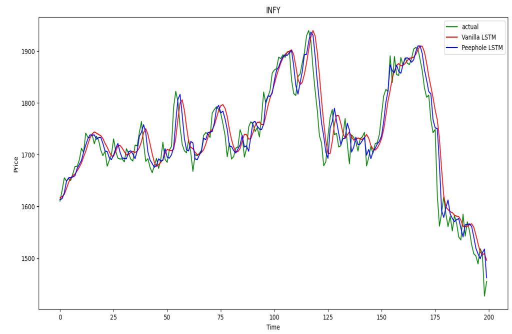

Fig. 11. Prediction plot for INFY

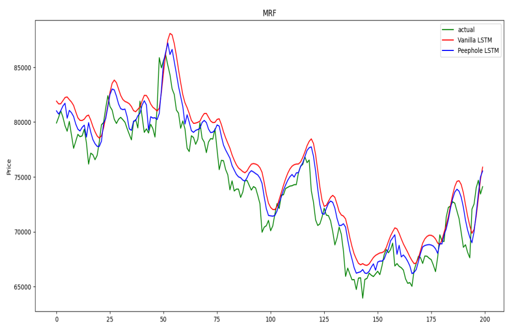

Fig. 9. Prediction plot for MRF

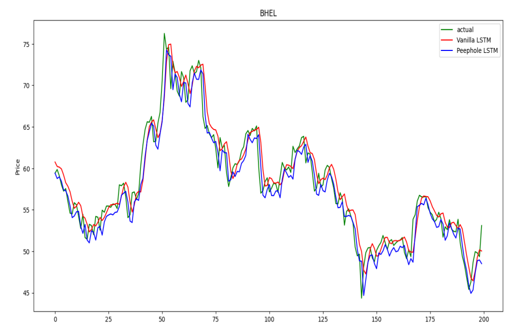

Fig. 12. Prediction plot for BHEL

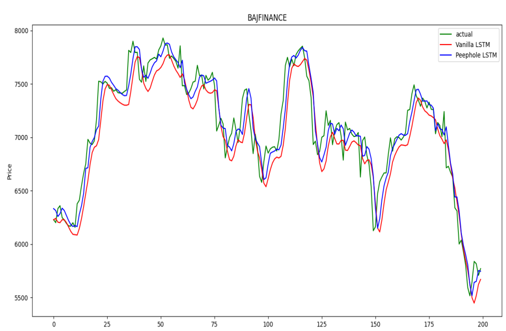

Fig. 10. Prediction plot for BAJFINANCE

Fig. 13. Prediction plot for SAIL

From the prediction plots in the figures 9, 10, 11, 12 and 13, it is clear that the blue curve (predictions by Peephole LSTM) is much closer than red curve (prediction by Vanills LSTM) to green curve (actual prices). This confirms the superiority of Peephole LSTM over Vanilla LSTM.

VII. Conclusion

This work presents a comparative study of the performance of two LSTM variants i.e. Vanilla LSTM and Peephole LSTM, for stock market price prediction. The algorithms are evaluated on five different stocks (MRF, BAJFINANCE, INFY, BHEL and SAIL) on their daily prices.

The results show that the Peephole LSTM variant performs better than the Vanilla LSTM. Peephole LSTM improved the average test RMSE by 6.04 % and the average test MAE by 9.85%.

There are a couple of exciting research avenues to pursue in the future. First, more data may be used to train the suggested model to see if adding more data improves the prediction performance. This can be done by training the models on hourly or even smaller time frames. Moreover, as the normalization is necessary to guarantee that the model generalizes well on unseen data, more advanced trainable normalization approaches may be applied. Finally, the effect of combining these two models can also be an exciting area of research.

References

[1] D. M. Q. Nelson, A. C. M. Pereira and R. A. de Oliveira, "Stock market's price movement prediction with LSTM neural networks," 2017 International Joint Conference on Neural Networks (IJCNN), 2017, pp. 1419-1426, doi: 10.1109/IJCNN.2017.7966019.

[2] S. Siami-Namini, N. Tavakoli and A. Siami Namin, "A Comparison of ARIMA and LSTM in Forecasting Time Series," 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), 2018, pp. 1394-1401, doi: 10.1109/ICMLA.2018.00227.

[3] S. A. Dwivedi, A. Attry, D. Parekh and K. Singla, "Analysis and forecasting of Time-Series data using S-ARIMA, CNN and LSTM," 2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), 2021, pp. 131-136, doi: 10.1109/ICCCIS51004.2021.9397134.

[4] Li, H., Shen, Y. and Zhu, Y., 2018, November. Stock price prediction using attention-based multi-input LSTM. In Asian conference on machine learning (pp. 454-469). PMLR.

[5] Fischer, T. and Krauss, C., 2018. Deep learning with long short-term memory networks for financial market predictions. European Journal of Operational Research, 270(2), pp.654-669.

[6] Mehtab, S., Sen, J. and Dutta, A., 2020, October. Stock price prediction using machine learning and LSTM-based deep learning models. In Symposium on Machine Learning and Metaheuristics Algorithms, and Applications (pp. 88-106). Springer, Singapore.

[7] Liu, Y., Su, Z., Li, H. and Zhang, Y., 2019, June. An LSTM based classification method for time series trend forecasting. In 2019 14th IEEE Conference on Industrial Electronics and Applications (ICIEA) (pp. 402-406). IEEE.

[8] Rokhsatyazdi, E., Rahnamayan, S., Amirinia, H. and Ahmed, S., 2020, July. Optimizing LSTM Based Network For Forecasting Stock Market. In 2020 IEEE Congress on Evolutionary Computation (CEC) (pp. 1-7). IEEE.

[9] Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. Neural computation, 9(8), pp.1735-1780.

[10] Gers, F.A., Schraudolph, N.N. and Schmidhuber, J., 2002. Learning precise timing with LSTM recurrent networks. Journal of machine learning research, 3(Aug), pp.115-143.