# Stock Price Prediction Using Recurrent Neural Networks

Lipin Guo
The University of Adelaide
Australia

**Abstract**

*In this paper, we aimed to perform a short-term stock price prediction task for IBM. The time series data was univariate. We compared three architectures of recurrent neural networks, Vanilla RNN, LSTM and GRU. GRU had the best performance with 0. 35% testing error rate. Compared to RNN, GRU can avoid gradient vanishing problem. Compared to LSTM, GRU performed better in long term memory.*

## 1. Introduction

Stock price prediction is one of forecasting problems involve the analysis of time series data. To predict the subsequent stock price, the models need to anlyze the history of prices to date. This chronological sequence of stock price is time series data. Forecasting problem can be classified as follows [1]

- Short term forecasting (prediction for seconds, minutes, days, weeks or months)
- Medium term forecasting (prediction for 1 or 2 years)
- Long term forecasting (prediction over 2 years)

In our case, our stock price prediction was short term forecasting. We would predict the subsequent day's stock price according to the history of prices. The time series data can either be univariate or multivariate. For various instances of time, univariate data contain information only about one specific stock while multivariate data contain stock price of multiple companies.

The existing methods for stock price prediction can be classified as follows [2]

- Fundamental Analysis
- Technical Analysis
- Time Series Forecasting

Fundamental analysis is most suitable for long term forecasting, which estimated a company share value by analysing its sales, earnings, profits and other economic factors [1]. Technical analysis is suitable for short term predictions. Moving Average (MA) is a commonly used algorithm in technical analysis, which can mitigate the impacts of random, short-term fluctuations on the price of a stock over a specified time frame. Time Series Forecasting is the third method. 'Analysis of time series data helps in identifying patterns, trends and periods or cycles existing in the data' [1]. This method involves two classes of algorithms:

- Traditional Machine Learning
- Deep Learning

The traditional machine learning including linear or non-linear models [1]. Linear models include Linear Regression, Auto Regression (AR), Auto Regressive Moving Average (ARMA), Auto-Regressive Integrated Moving Average (ARIMA) and its variants [3][4][5]. These models use some predefined equations to fit a mathematical model to a univariate time series. The non-linear models include ARCH, GARCH [4].

Deep learning algorithms also can regard as non-linear function approximator, which identify hidden patterns and underlying dynamics in time series data automatically. They include Multi-layer Perceptron (MLP), Recurrent Neural Networks (RNN), CNN (Convolutional Neural Network), Long Short Term Memory (LSTM), Gated Recurrent Unit (GRU) etc [6].

In the last few years, recurrent neural networks are popular in stock price prediction especially using multivariate time series data. In 1996, Raman and Jameel proposed to use RNNs in financial analysis of multiple stock market returns [7]. In 2015, different architectures of neural networks had been used for multivariate financial time series analysis [8]. In 2017, we can see the comparison among RNN, LSTM and CNN for multivariate stock price prediction [1]. In 2018, Minh et al. proposed that two-stream GRU outperformed state-of-the-art models [9]. In 2022, Vikas et al. proposed an integrated approach towards stock price prediction using LSTM algorithm [10]. In 2022, Bhavani et al. classified that GRU outperforms LSTM in terms of performance in multivariate stock price prediction [11]. Also, some researchers used LSTM to forecast Adjusted Closing Price, Movement Price [12][13].

In this paper, our stock price prediction was short term forecasting. Our time series data would be univariate using IBM stock as there is a blank research area for univariate stock prediction compared to multivariate stock prediction using deep learning algorithms. We compared three architectures of recurrent neural networks, RNN, LSTM

and GRU. They have been identified to forecast the next day's stock price of IBM based on its shutting cost of previous 60 days market prices in this paper. We are focusing on the Close Price of the stock as Close value timings are fixed as because of AMO (After Market Orders) and PMO (Post Market Orders) compared to Open Price timings [14].

## 2. Methodology

The data set is the stock price of the only one company IBM for the period of January 2018 to November 2022. It includes information like date, open, high, low, close, adjusted close, volume of stock sold in each day. The data length is 1230, 80% for training and 20% for testing. The data was subjected to normalization and was mapped to a range of 0 to 1 to unify the data range.

The work is based on a sliding window approach for a short-term future prediction. The window was fixed with overlap of 60 days' information and prediction was made for the 61st day in future. The best window length was identified by calculating the error for various window size.

All the models were trained for 50 epochs by changing the depth of layers for fine-tuning. The discrepancy between output $O_t$ and the desired target $y_t$ is evaluated by the objective function (1) across all the T time steps as [15]

$$L = \frac{1}{T}\sum_{t=1}^{T} l(y_t, o_t) \quad (1)$$

If the loss function $l(y_t, o_t)$, which uses Mean Square Error (MSE), of current epoch was smaller than the value in previous epoch, the weight matrices for that epoch would be updated. After training, these models were tested and the one with lowest MSE is the final model for prediction.

We have used three different deep learning architectures, RNN, LSTM and GRU for this work.

### 2.1. RNN

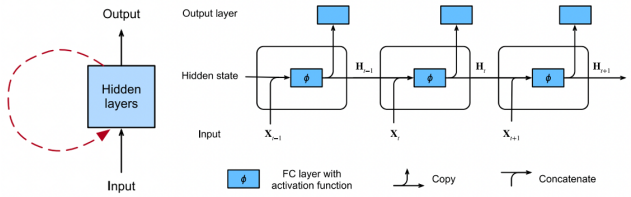


Figure 2.1: The architecture of RNN [15]

In this paper, the RNN we discussed was Vanilla RNN. As the left part of Figure 2.1 shown, unlike other feed forward neural networks, RNN captures the dynamics of sequences through recurrent connections, which can be thought of as loops in a network of nodes. The feedforward nature of neural networks that makes the order of recurrent connections computation explicit. As the right part of Figure 2.1 shown, a recurrent neural network is unrolled across time steps (or sequence steps), with the same underlying parameters applied to each step. While standard connections are applied synchronously to propagate the activations of each layer to subsequent layers at the same time step, recurrent connections are dynamic, passing information through adjacent time steps [15]. The hidden state (2) and output (3) in RNN at each time step are [15]

$$H_t = f(X_t, H_{t-1}, W_h) \quad (2)$$
$$O_t = g(H_t, W_o) \quad (3)$$

where $f$ and $g$ are transformations of the hidden layer and the output layer respectively. For equation (2), at any time step $t$, the computation of the hidden state would concatenate the input $X_t$ (information about 60 days' stock price) at the current time step and the hidden state $H_{t-1}$ at the previous time step [15]. Then, feeding the result into a fully connected layer with Tanh activation function. The output of such a fully connected layer is the hidden state of the current time step $t$. For equation (3), at any time step $t$, $h_t$ will also be fed into the fully connected output layer to compute the output $O_t$ (stock price prediction for 61st day) [15].

### 2.2. LSTM

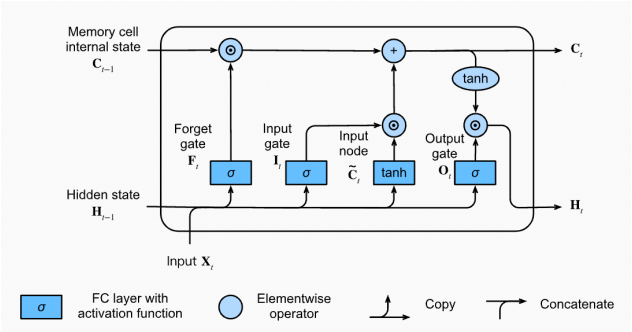


Figure 2.2: The Architecture of LSTM [15]

LSTM is recurrent neural network. The key to LSTMs is the cell state (or memory cell internal state), the horizontal line running through the top of the Figure 2.2. The cell state has only a few minor linear interactions, so it is easy for information to flow along it without change. As Figure 2.2 shows, there are three kinds of gates in LSTM, including input gate, forget gate and output gate. Gates can control the removal or addition of information to the cell state. They are 'composed out of a sigmoid neural net layer and a pointwise multiplication operation' [16]. The sigmoid layer outputs a number between 0 and 1 describing how much of each component should pass [16]. The value of 0 means no information can go through, while the value of 1 means let all the information go through. The computation of forget gate layer (4), input gate layer (5) and output gate layer (6) are defined as below [15]

$$F_t = \sigma(W_f \cdot [H_{t-1}, X_t] + b_f) \quad (4)$$
$$I_t = \sigma(W_i \cdot [H_{t-1}, X_t] + b_i) \quad (5)$$
$$O_t = \sigma(W_o \cdot [H_{t-1}, X_t] + b_o) \quad (6)$$
$$\widetilde{C}_t = tanh(W_c \cdot [H_{t-1}, X_t] + b_c) \quad (7)$$

$$C_t = F_t * C_{t-1} + I_t * \widetilde{C}_t \quad (8)$$

$$O_t = \sigma(W_o \cdot [H_{t-1}, X_t] + b_o) \quad (9)$$

$$H_t = O_t * tanh(C_t) \quad (10)$$

The forget gate layer (left in Figure 2.2), which is a sigmoid layer, determines whether to 'keep the current value of the memory or flush it' [15]. In this layer, the hidden state $H_{t-1}$ at previous time step and input $X_t$ (information about 60 days' stock price) in the current time step will be mapped using sigmoid function. Then the output $F_t$ was a value between 0 and 1. $F_t = 1$ determines completely keep the values (or information) in the cell state $C_{t-1}$ while $F_t = 0$ determines getting rid of the values (or information) in the cell state $C_{t-1}$ completely. Then, LSTM decides which new information store in the cell state. The input gate layer (also a sigmoid layer) $I_t$ decides which values we'll update. Later, the input node $\widetilde{C}_t$ (which is tanh layer) outputs a value in range (-1,1), defined as equation (7). The computation of $I_t$ and $\widetilde{C}_t$ will be updated to old cell state $C_{t-1}$. After that, as the equation (8) showed, the old cell state $C_{t-1}$ would be new cell state $C_t$ by the addition of two values, one values is the multiplication result of $F_t$ and $C_{t-1}$, another value is the multiplication result of $I_t$ and $\widetilde{C}_t$. Finally, the model decides what to output. As equation (9) showed, the output gate layer is a sigmoid layer, which outputs a value $O_t$ (stock price prediction for 61st day). This value $O_t$ will be multiplied by the output value of tanh layer (which put the new cell state through it). This result will be new hidden state $H_t$ as equation (10) showed.

### 2.3. GRU

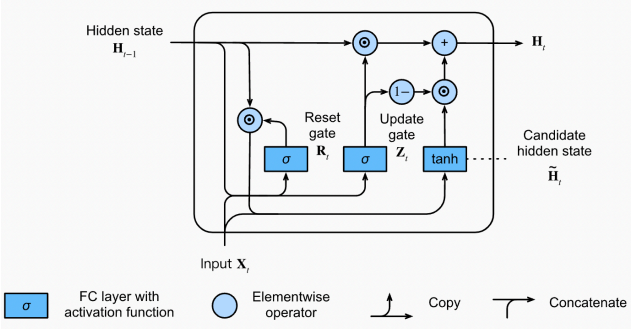


Figure 2.2: The Architecture of GRU [15]

GRU is also recurrent neural network. Compared to LSTM, GRU only have two gates: reset gate and update gate, as Figure 2.2 showed. The reset gate layer and update gate layer are fully connected layers with sigmoid function, which the outputs in the interval of (0, 1). The computation of reset gate $R_t$ and update gate $Z_t$ for each time step are as follow equation (11) (12) respectively [15]

$$R_t = \sigma(X_t W_{xr} + H_{t-1} W_{hr} + b_r) \quad (11)$$

$$Z_t = \sigma(X_t W_{xz} + H_{t-1} W_{hz} + b_z) \quad (12)$$

$$\widetilde{H}_t = tanh(X_t W_{xh} + R_t \odot H_{t-1}) W_{hh} + b_h) \quad (13)$$

$$H_t = Z_t \odot H_{t-1} + (1 - Z_t) \odot \widetilde{H}_t \quad (14)$$

The hidden state $H_{t-1}$ at previous time step and input $X_t$ (information about 60 days' stock price) in the current time step will be mapped using sigmoid function with their own weighs $W_{hr}$ and $W_{xr}$ respectively. The output of reset gate $R_t$ determines how much of the previous state we might still want to remember [15]. The output of update gate $Z_t$ controls how much of the past information from the old state $H_{t-1}$ needs to be passed along to the future. The candidate hidden state $\widetilde{H}_t$ is the composed out of a fully connected layer (with tanh activation function) and an elementwise product operation, indicated by equation (13). The influence of the previous states can be reduced with elementwise multiplication of $R_t$ and $H_{t-1}$. When the output of reset gate $R_t$ is close to 1, a vanilla RNN will be recover. When $R_t$ is close to 0, $\widetilde{H}_t$ is the 'result of an MLP with $X_t$ as input. Any pre-existing hidden state is thus reset to defaults' [15]. According to equation (14), we can get the new state $H_t$. When the update gate $Z_t$ is close to 1, the model will retain the old state and ignore the information from $X_t$. When $Z_t$ is close to 0, the new state $H_t$ will be approximate to $\widetilde{H}_t$.

### 2.4. Type of Recurrent Neural Network Architectures

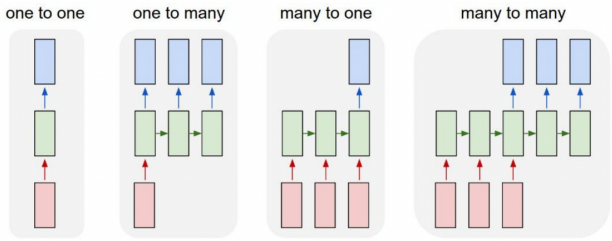


Figure 2.4: Different Types of RNN architectures [17]

Recurrent Neural Network architectures have different types to suit different tasks, including one-to-one, one-to-many, many-to-one, many-to-many [17]. Our task was to use previous 60 day's stock price to predict the 61st day's price. Our models received a sequence of time series data (previous 60 day's stock price) and produce a single output (prediction of the 61st day's price). Therefore, we used many-to-one recurrent neural network architectures. RNN, LSTM and GRU would have same type based on our task.

## 3. Experiment and Analysis

The aim of experiment design is to compare the performance among RNN, LSTM and GRU evaluated by the test error rate (MSE) using IBM stock dataset and find the best model with lowest test error rate.

To conduct a symmetric comparison, we compared the models based on the condition. To find a proper condition, we tried out different combinations of learning rate (0.01, 0.001), epoch numbers (20, 50, 100), batch size (8, 256) and optimizer (SGD, Adam). As our task target was to predict the stock price of subsequent day correctly, the discrepancy between output $O_t$ and the desired target $y_t$ was evaluated by the objective function (using equation (1) in Section 2 Methodology part) across all the T time steps. And we would like to minimize the loss function, which means the loss decrease and loss curve converge faster. The best hyperparameters in our experiment for all the models are

- Learning rate = 0.01
- Epoch number = 50
- Batch size = 256
- Optimizer = Adam

Based on the best combination of parameters, we discussed the performance of RNN, LSTM and GRU.

### 3.1. Gradient Vanishing


Figure 3.1: Stock Price Testing Prediction Curve: (a) RNN; (b) LSTM; (c) GRU. When the depth = 5, the prediction results of three models are different.

| Models | Window Length | Depth | Test error rate |
|---|---|---|---|
| RNN | 60 | 5 | - |
| LSTM | 60 | 5 | 2.05% |
| GRU | 60 | 5 | 1.51% |

Table 3.1: Test Error Rate of RNN, LSTM, GRU with different depth

From Table 3.1, we can see that GRU had the best performance with lowest test error rate whether the depth was 5. LSTM was the worser one, and the RNN was the worst and failed to predict. As Figure 3.1(a) showed, we can see that RNN suffer gradient vanishing problem when the depth was 5, while LSTM and GRU did not suffer as Figure 3.1 (b) and (c) showed respectively.

This was because the backpropagation in RNN, which called Backpropagation Through Time [18]. The unrolled RNN was essentially a feed-forward neural network with the special property that the same parameters are repeated throughout the unrolled network, appearing at each time step. Then, as in any feedforward neural network, we could apply the chain rule to backpropagate the gradients through the unrolled network. The gradients for each parameter must be summed over all locations in the unrolled network where the parameter occurs. Because the sequence can be quite long, gradient problems can be caused by computation (excessive memory) and optimization (numerical instability) during gradient backpropagation [15]. Gradient had dramatic changes in a small area of error surface, sometimes very large, sometimes very small, which will cause two kinds of problems:

- Gradient exploding problem happens during backpropagation through time, where large error gradients accumulate and result in very large updates to neural network model weights. And it can be solved by can use truncated backpropagation through time or gradient clipping to prevent exploding gradient.
- Gradient vanishing problem is encountered during backpropagation through time, where weights receive an update proportional to the partial derivative of the error function with respect to the current weight. The gradient will be vanishingly small, resulting no change of weights and stopping the recurrent neural network form further training.

As Figure 3.1(b) showed, the gradient vanishing problem could be handled by LSTM, which was the first successful techniques for addressing vanishing gradients. The reason was LSTM and RNN had different ways to deal with memory. In RNNs, they can use their internal memory to process arbitrary input sequence. The memory (previous output/immediate past) in the hidden state at each time step would be formatted after updating weights. The influence of previous weight would disappear. LSTM replaces every ordinary recursive node of RNN with memory cells. Each memory cell contains an internal state, a node with self-connected loop edges of fixed weight, ensuring that gradients through cell state instead of hidden state can span multiple time steps without vanishing [15].

As Figure 3.1(c) showed, GRU also can avoid gradient vanishing problem. GRU has two gates as Section 2.3 mentioned. Reset gate help capture short-term dependencies in sequences and update gate help capture long-term dependencies in sequences. The gradient vanishing problem is a long-term dependencies problem, which can be handled by GRU.

Considering the gradient vanishing problem, we only compared compare the performance among LSTM and GRU with the changing of depth and slicing window length.

### 3.2. Comparative Analysis of LSTM and GRU

| Models | Window Length | Depth | Test error rate |
|---|---|---|---|
| LSTM | 40 | 3 | 0.89% |
| LSTM | 60 | 3 | 1.70% |
| LSTM | 60 | 5 | 2.05% |
| GRU | 40 | 3 | 0.73% |
| GRU | 60 | 3 | 0.35% |
| GRU | 60 | 5 | 1.51% |

Table 3.2: Test Error Rate of LSTM, GRU

From Table 3.2, we can see that GRU had the best performance of 0.35% test error. However, when the number of stacked recurrent layers (or depth of architectures) increased and the window length became shorter, the performance of LSTM and GRU became worser. In general, GRU performed better than LSTM.

For comparatively long-term memory, as LSTM and GRU had different architectures (as Section 2 Methodology part mentioned) and they used different ways to remember information, thus they had different performance.

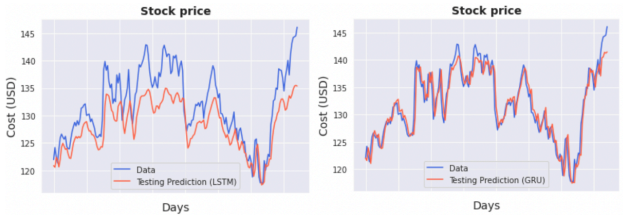


Figure 3.2: Stock Price Testing Prediction Curve: (a) LSTM with depth = 3, length=60 (left); (b) GRU with depth = 3, length=60 (right).

From Figure 3.2 (a) and (b), the prediction curve of prediction curve of GRU was with great approximation to the original stock price. However, LSTM tried to remember the trend, but it failed to predict as the predicted value was far away from original data.

One distinction of LSTM from RNN is that LSTM has additional cell state $C_t$ and intuitively it can be thought of as $C_t$ stores long-term information. If the forget gate was set to 1, and the input gate was set to 0, then the information of the cell state will always be preserved over many recurrent time steps. For a RNN, it was much harder to preserve information in hidden states in recurrent time steps by just making use of a single weight matrix. However, GRU was effective than LSTM. That's because in GRU if the old information was not formatted then the new information won't be updated. The reset gate in GRU decided whether to drop previous information or not. If there was no value to format in reset gate, then the update gate would open and decide how much past information should be updated. As GRU had two gates compared to three gates in LSTM, it needs less parameters and can avoid overfitting.

In short, GRU has the best performance with the lowest test error rate of 0.35% (shown in Table 3.2) and great stock prediction result (shown in Figure 3.2(b)). It can avoid gradient vanishing problem which often happened in RNN. When the model became deeper and window length increased, it performed better than LSTM as well.

## 4. Code Implementation

As for the code, please refer to https://github.com/LipinGuo/RNN. The code implementation was separated into various jupyter notebooks according to different types of experiments.

## 5. Conclusion

In conclusion, our task was to predict IBM's stock price daily which is a short-term forecasting problem, and the data is univariate time series data. We tried three different architectures of RNNs, vanilla RNN, LSTM and GRU. We performed fine-tuning to try different combination of learning rate, batch size, epoch number and optimizer for all the models, we found that best hyperparameters to minimize objective function is

- Learning rate = 0.01
- Epoch number = 50
- Batch size = 256
- Optimizer = Adam

The best model we found was GRU with 0. 35% testing error rate. This model has lowest test error rate and great prediction curve compared to other models. Compared to RNN, GRU can avoid gradient vanishing problem. Reset gate help capture short-term dependencies in sequences and update gate help capture long-term dependencies in sequences. Compared to LSTM, GRU performed better in long time memory task, which used two gates to obtain similar function of LSTM and more effective with less parameters.

In future, we can:

1) According to the limitation of data, we could try multivariate data and larger dataset. Our models only consider univariate time series, so these models fail to identify dependencies among various stocks. Furthermore, a model identified for one series may not be suitable for another. Thus, we may try multivariate data in the future. As IBM dataset only consists of 1230 dataset, if we changed to a larger dataset, the conclusion would be influenced.
2) To address the gradient vanishing problem, we can try other techniques, such as Clockwise RNN [20], SCRN [21].
3) As this data was a small dataset, traditional algorithms as we mentioned in Section 1 may have a good performance. In this paper, our best model GRU was based on comparison with RNN and LSTM, which were belong to deep leaning algorithms. We could try to compare GRU with traditional algorithms. For example, Youness et al. (2022) compared the performance of LSTM and ARIMA for a univariate time series, then classified that ARIMA can outperform deep learning ones since they are very simple to use especially for linear univariate datasets [23].
4) The task we performed in this paper was short-term forecasting. We could try to use GRU to

perform medium-term forecasting (prediction for 1 or 2 years) or long-term forecasting (prediction over 2 years) and compare its performance in more complex task.

## References


[1] S. Selvin, R. Vinayakumar, E. A. Gopalakrishnan, V. K. Menon and K. P. Soman, "Stock price prediction using LSTM, RNN and CNN-sliding window model," *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2017, pp. 1643-1647, doi: 10.1109/ICACCI.2017.8126078.

[2] A. V. Devadoss and T. A. A. Ligori, "Forecasting of stock prices using multi layer perceptron," *Int J Comput Algorithm*, vol. 2, pp. 440–449, 2013.

[3] J. G. De Gooijer and R. J. Hyndman, "25 years of time series forecasting," International journal of forecasting, vol. 22, no. 3, pp. 443–473, 2006.

[4] V. K. Menon, N. C. Vasireddy, S. A. Jami, V. T. N. Pedamallu, V. Sureshkumar, and K. Soman, "Bulk price forecasting using spark over nse data set," in International Conference on Data Mining and Big Data. Springer, 2016, pp. 137–146.

[5] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, Time series analysis: forecasting and control. John Wiley & Sons, 2015.

[6] Y. Bengio, I. J. Goodfellow, and A. Courville, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[7] J. Roman and A. Jameel, "Backpropagation and recurrent neural networks in financial analysis of multiple stock market returns," *System Sciences*, 1996., Proceedings of the Twenty-Ninth Hawaii International Conference on, vol. 2. IEEE, 1996, pp. 454–460.

[8] G. Batres-Estrada, "Deep learning for multivariate financial time series," *ser. Technical Report*, Stockholm, May 2015.

[9] D. Lien Minh, A. Sadeghi-Niaraki, H. D. Huy, K. Min and H. Moon, "Deep Learning Approach for Short-Term Stock Trends Prediction Based on Two-Stream Gated Recurrent Unit Network," *in IEEE Access*, vol. 6, pp. 55392-55404, 2018, doi: 10.1109/ACCESS.2018.2868970.

[10] K. Vikas, K. Kotgire, B. S. P. Reddy, A. S. K. Teja, H. V. Reddy and S. Salvadi, "An Integrated Approach Towards Stock Price Prediction using LSTM Algorithm," *2022 International Conference on Edge Computing and Applications (ICECAA)*, 2022, pp. 1696-1699, doi: 10.1109/ICECAA55415.2022.9936082.

[11] A. Bhavani, A. V. Ramana and A. S. N. Chakravarthy, "Comparative Analysis between LSTM and GRU in Stock Price Prediction," *2022 International Conference on Edge Computing and Applications (ICECAA)*, 2022, pp. 532-537, doi: 10.1109/ICECAA55415.2022.9936434.

[12] Y. -X. Luo and Y. Ji, "Prediction of the Stock Adjusted Closing Price Based On Improved PSO-LSTM Neural Network," *2022 International Conference on Machine Learning and Cybernetics (ICMLC)*, 2022, pp. 43-48, doi: 10.1109/ICMLC56445.2022.9941330.

[13] 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.

[14] Reshma.R., Usha Naidu, V.Sathiyavathi, and L.SaiRamesh. "Stock Market Prediction Using Machine Learning Techniques." *Advances in Parallel Computing Technologies and Applications*, vol.40, pp.331-340, (2021)

[15] Zhang, Aston and Lipton, Zachary C. and Li, Mu and Smola, Alexander J., "Dive into Deep Learning", arXiv:2106.11342, 2021

[16] https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[17] https://github.com/cs231n/cs231n.github.io/blob/master/rnn.md

[18] Werbos, PJ, 1990, 'Backpropagation through time: what it does and how to do it', *Proceedings of the IEEE*, 78(10), 1550–1560.

[19] Kanai, S, Fujiwara, Y, Iwamura, S, 2017, 'Preventing Gradient Explosions in Gated Recurrent Units', *Advances in Neural Information Processing Systems 30 (NIPS 2017)*

[20] Jan Koutník, Klaus Greff, Faustino Gomez, Jürgen Schmidhuber, 'A Clockwork RNN', *Neural and Evolutionary Computing*, 2014

[21] Tomas Mikolov, Armand Joulin, Sumit Chopra, Michael Mathieu, Marc'Aurelio Ranzato, 'Learning Longer Memory in Recurrent Neural Networks', 2015

[22] Quoc V. Le, Navdeep Jaitly, Geoffrey E. Hinton, 'A Simple Way to Initialize Recurrent Networks of Rectified Linear Units', 2015

[23] J. Youness and M. Driss, "LSTM Deep Learning vs ARIMA Algorithms for Univariate Time Series Forecasting: A case study," *2022 8th International Conference on Optimization and Applications (ICOA),* 2022, pp. 1-4, doi: 10.1109/ICOA55659.2022.9934119.