Predicting Bitcoin Price Using Recurrence Neural Network

Zhelong Chen
University of Guelph
Guelph, Canada
zhelong@uoguelph.ca

***Abstract* — In this report, a prediction method for bitcoin's pricing using Recurrence neural network is described. Recurrence neural network was a concept introduced in CIS4020 lecture. The main goal of this report is to apply the knowledge we learn throughout the semester and apply them in real world area of applications. In the report below, I will describe how the prediction performs with a given dataset containing historical price.**

Index terms – cryptocurrency, bitcoin, data science, bitcoin prediction, neural network

I. Introduction

Crypto currency is a tradable digital form of money, built on the tech stack of blockchain and can only be accessed online. Similar with stock markets, crypto markets are also influenced by many factors, some comes with risks.

Bitcoin is the longest running and most well known cryptocurrency, first released as open source in 2009 by the anonymous Satoshi Nakamoto[1]. Bitcoin serves as a decentralized medium of digital exchange, with transactions verified and recorded in a public distributed ledger (the blockchain) without the need for a trusted record keeping authority or central intermediary. Transaction blocks contain a SHA-256 cryptographic hash of previous transaction blocks, and are thus "chained" together, serving as an immutable record of all transactions that have ever occurred[1].

Predicting future trends is a popular research field in data science study, for example prediction of covid trend, real estate price, crypto currency and many others. There can be many beneficial impacts with the correct prediction. In specific, people/groups can increase their asset values with the right prediction containing high amount of accuracy on bitcoin trends. However, it remains a challenge that no models can perfectly predict the trend with many uncertain variables such as political impact, big tech companies' policy on bitcoin transaction and other economic changes.

Recurrence Neural Network

A Recurrent Neural Network (RNN) is a type of neural network well-suited to time series data. RNNs process a time series step-by-step, maintaining an internal state from time-step to time-step [2].

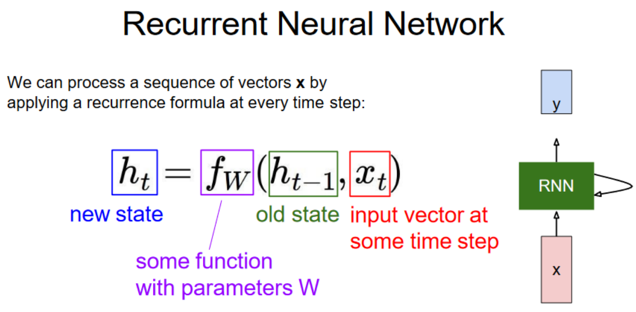

Fig.0. General Idea of Recurrent Neural Network, image by Andrej Karpathy

LSTM, short for Long short-term memory is a recurrent learning system that prevents backpropagated errors from vanishing or explode. LSTM was created by Hochreiter & Schmidhuber (1997) [3] and later developed and popularized by many researchers. There exists an issue with some basic NNs, such as vanilla neural network, where the back propagation error can lead to 0 or infinity. The backpropagated errors in LSTM can pass through layers unlimited times backwards limited data loss.

On Prediction, we can predict Bitcoin using technique on the specific subject that we wanted. Example, we want to predict only by the signal or the price, or we can predict just for current day or next day close value based on Long Short Term Memory (LSTM) [9].

Bitcoin Historical Dataset [4].

The dataset I am using contains detailed bitcoin pricing information [4]. It is a CSV files for bitcoin exchanges for the time period of 2012-01-01 to 2021-03-31, with 60 seconds updates of Open, High, Low, Close. The price is in USD currency. The Volume represents bitcoin currency, and weighted bitcoin price. The features of the dataset include (Fig. 1):

Timestamp: Start time of time window (60s window), in Unix time

Open: Open price at start time window

High: High price within time window

Low: Low price within time window

Close: Close price at end of time window

Volume_(BTC): Volume of BTC transacted in this window

Volume_(Currency): Volume of corresponding currency transacted in this window.

Weighted_Price: Volume Weighted Average Price.

Timestamp	Open	High	Low	Close	Volume_(BTC)	Volume_(Currency)	Weight
1325317920	4.39	4.39	4.39	4.39	0.455581	2.0	

Fig. 1. Dataset features

II. PREPARATION

A. Cleaning the dataset

The raw dataset contains NaN values in many entries and the timestamp is in Unix time (Fig. 2).

```
data.head()
```

	Timestamp	Open	High	Low	Close	Volume_(BTC)	Volume_(Currency)	Weighted_Price
0	1325317920	4.39	4.39	4.39	4.39	0.455581	2.0	4.39
1	1325317980	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	1325318040	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	1325318100	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	1325318160	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Fig. 2. Raw Dataset visualization

A few data cleaning methods are used, including convert the time stamp from Unix time to minutes, clear the null rows by averaging values along the time stamp in day units. The datasets are a lot more interpretable and practical after the clean up (Fig. 3). The clean up also reduced the dataset by 1440 times, provided a faster running time.

	Timestamp	Open	High	Low	Close	Volume_(BTC)	Volume_(Currency)	Weighted_Price
0	2011-12-31	4.465000	4.482500	4.465000	4.482500	23.829470	106.330084	4.471603
1	2012-01-01	4.806667	4.806667	4.806667	4.806667	7.200667	35.259720	4.806667
2	2012-01-02	5.000000	5.000000	5.000000	5.000000	19.048000	95.240000	5.000000
3	2012-01-03	5.252500	5.252500	5.252500	5.252500	11.004660	58.100651	5.252500
4	2012-01-04	5.200000	5.223333	5.200000	5.223333	11.914807	63.119577	5.208159

Fig. 3. Dataset visualization after cleaning up

The dataset has 3376 rows by 8 columns after cleaning up, where each column represents a feature, and each row represents the bitcoin's features of the day.

```
data.shape
```

```
(3376, 8)
```

B. Visualize the dataset

To prepare for prediction and have a better understanding of the dataset, A plot of bitcoin's historical price is plotted. It measures the time in days vs price in weighted average. (Fig. 4.).

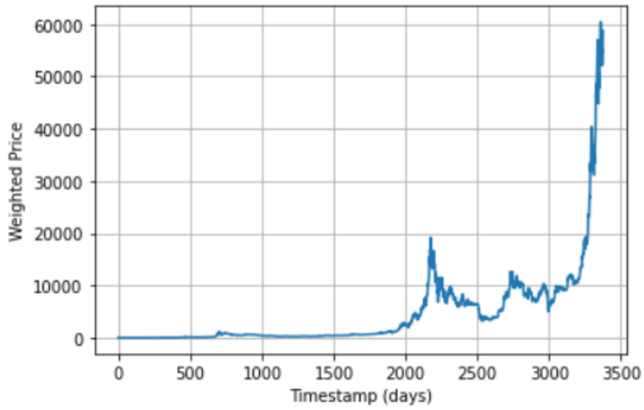

Fig.4. Bitcoin's price from 2012-01 to 2021-03

From the graph, we can see that the price remains low for the first few years and curved up with a steep incline in 2017 and 2021. It is not easy to predict this type of datasets because there are many uncertainties that causing local maxima and minima with global maxima and global minima.

A few more plots are plotted for us to observe any interesting relations, and to help understand more about the dataset with further predictions:

From Fig.5. We observe that the as bitcoin's price increases, the transaction volume decreases. This presented an inverse proportional relationship between them. The times that when bitcoin's transaction volume was large, was in the early few years when the price was not high.

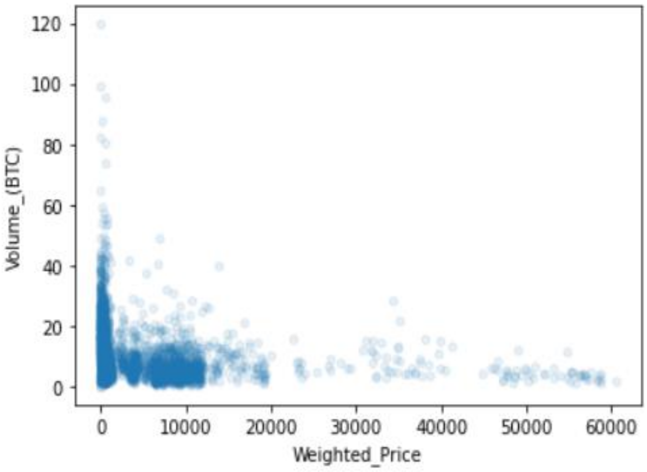

Fig. 5. Bitcoin Price vs Transaction Volume

From Fig. 6. We observe that as bitcoin price increases, the transaction currency (USD) increases. It showed a linear proportional relationship between bitcoin price and transaction currency.

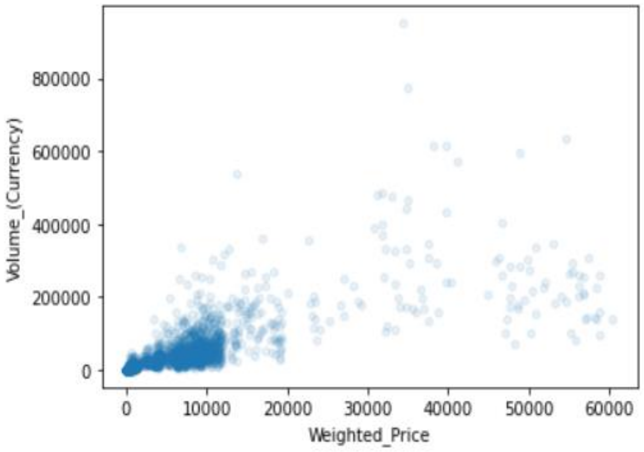

Fig. 6. Bitcoin Price vs Transaction Currency.

C. *Overview of Recurrence Neural Network*

The RNN structure will be implemented in Keras, a Python machine learning library. Each neural network cell takes one data input and one hidden state which is passed from a one-time step to the next. An example of RNN cell looks as in Fig. 7.

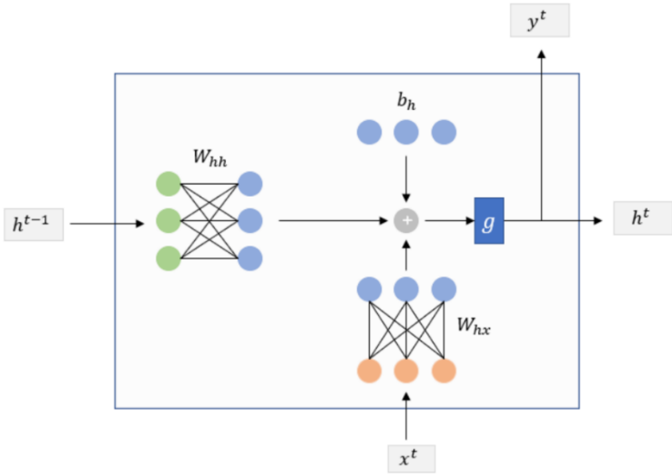

Fig. 7. The flow of data and hidden state inside the RNN cell implementation in Keras. Image by Mohit Mayank[5]

The formula of an RNN cell is provided below:

$$h^t = g(W_{hh}h^{t-1} + W_{hx}x^t + b_h)$$
$$y^t = h^t$$

Where h^t and h^{t-1} are the hidden states from time t and t-1. There are 2 weight matrices W_{hh} and W_{hx} that represents as an internal layer NN, and the bias term b_h. g() is the activation function in the formula, it's default is 'tanh'.

D. *General concept on the LSTM Model [6][7]*

Long-short term memory is a useful network for modeling time-series data; hence this is the reason it was used to predict the bitcoin pricing – time-series data.

As shown in Fig. 7, recurrent neural networks have the form of chain like structure with a simple layer in each cell. LSTM has a more complex model with four different layers.

The variables below have the following definition in the equation:

1). x: input information scaling

2). σ: Sigmoid layer, outputting between 0 and 1

3). h: output of a LSTM unit

4). c: memory of a LSTM unit

5). W: weight function

6). b: bias term

The first layer is considered as a filter layer, where it decides what information to keep and what to discard. It reads in the input pair h_{t-1} and x_t and output a number between 0 and 1, where the value has a higher chance to be discarded if it is closer to 0 and vice versa.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

The second layer is to decide what information is stored in the cell. A tanh() layer creates a vector of values, that could be added to the cell. Then it is decided by another layer to decide whether it will be updated.

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C$$

Afterwards it updates the old cell state, C_{t-1} into the new cell state, C_t.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

In the last layer it is decides what is the output value. It puts the cell state through tanh() to produce a value, then a sigmoid layer applies on top to filter the part to output.

$$o_t = \sigma\left(W_o \ [h_{t-1}, x_t] \ + \ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

LSTM layer is implemented using Keras library.

III. METHODS

A. *Data Scaling & normalization*

The dataset containing price is split into training set and testing set. It was split with respect to the last 60 days, which means training set has the data from January 2012 to January 2021 and testing dataset contained prices from Jan 2021 to March 2021. The dataset was split in this way because if we were using the traditional 80/20 split, there could be a chance that cause errors and produce high amount of variance when we are predicting. It was derived from Fig. 4 that bitcoin price skyrocketed in mid-2020.

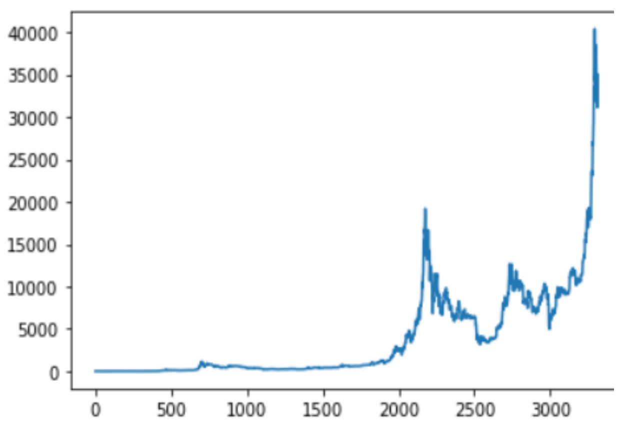

Fig.8. Training data

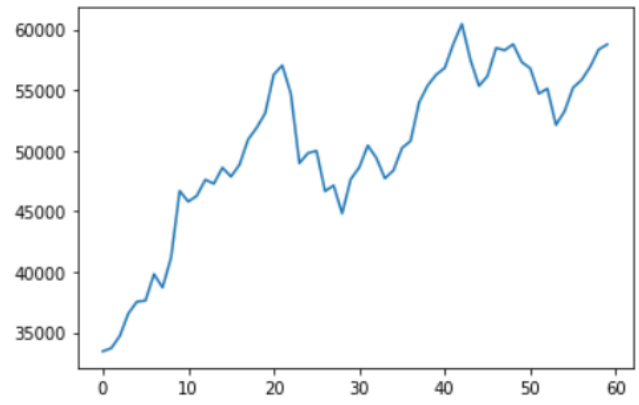

Fig.9. Testing data

The training data was scaled using min-max scaling, produced a set of data in between 0 and 1 for better outlier detection.

B. *Building the model*

Four layers of neural network model is applied, and result is shown in Fig.10.

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_7 (LSTM)                (None, 60, 70)            20160

dropout_4 (Dropout)          (None, 60, 70)            0

lstm_8 (LSTM)                (None, 60, 80)            48320

dropout_5 (Dropout)          (None, 60, 80)            0

lstm_9 (LSTM)                (None, 60, 90)            61560

dropout_6 (Dropout)          (None, 60, 90)            0

lstm_10 (LSTM)               (None, 60, 100)           76400

dropout_7 (Dropout)          (None, 60, 100)           0

=================================================================
Total params: 206,440
Trainable params: 206,440
Non-trainable params: 0
```

Fig.10. Model Summary

The result from the model fit is displayed below (Fig. 11). The loss column represents the training loss and val_loss represents the validation loss. We observed that there is an insignificant amount of training loss and small amount of validation loss, much higher relatively to training loss.

```
Epoch 1/20
1465/1465 [==============================] - 132s 83ms/step - loss: 0.0071 - val_loss:
Epoch 2/20
1465/1465 [==============================] - 115s 79ms/step - loss: 0.0068 - val_loss:
Epoch 3/20
1465/1465 [==============================] - 122s 83ms/step - loss: 0.0066 - val_loss:
Epoch 4/20
1465/1465 [==============================] - 115s 79ms/step - loss: 0.0067 - val_loss:
Epoch 5/20
1465/1465 [==============================] - 120s 82ms/step - loss: 0.0068 - val_loss:
Epoch 6/20
1465/1465 [==============================] - 129s 88ms/step - loss: 0.0066 - val_loss:
Epoch 7/20
1465/1465 [==============================] - 126s 86ms/step - loss: 0.0066 - val_loss:
Epoch 8/20
1465/1465 [==============================] - 126s 86ms/step - loss: 0.0068 - val_loss:
Epoch 9/20
1465/1465 [==============================] - 115s 79ms/step - loss: 0.0068 - val_loss:
Epoch 10/20
1465/1465 [==============================] - 125s 85ms/step - loss: 0.0066 - val_loss:
Epoch 11/20
1465/1465 [==============================] - 129s 88ms/step - loss: 0.0065 - val_loss:
Epoch 12/20
1465/1465 [==============================] - 119s 81ms/step - loss: 0.0068 - val_loss:
Epoch 13/20
1465/1465 [==============================] - 116s 79ms/step - loss: 0.0065 - val_loss:
Epoch 14/20
1465/1465 [==============================] - 128s 87ms/step - loss: 0.0067 - val_loss:
Epoch 15/20
1465/1465 [==============================] - 127s 87ms/step - loss: 0.0067 - val_loss:
Epoch 16/20
1465/1465 [==============================] - 126s 86ms/step - loss: 0.0065 - val_loss:
Epoch 17/20
1465/1465 [==============================] - 125s 85ms/step - loss: 0.0065 - val_loss:
Epoch 18/20
1465/1465 [==============================] - 125s 85ms/step - loss: 0.0066 - val_loss:
Epoch 19/20
1465/1465 [==============================] - 116s 79ms/step - loss: 0.0065 - val_loss:
Epoch 20/20
1465/1465 [==============================] - 121s 82ms/step - loss: 0.0064 - val_loss:
```

Fig. 11. loss data

To help visualize the data loss and have a better interpretation, a graph is created (Fig. 12). It shows that both losses are relatively small, however validation loss is much greater than training loss. A summary we can make is that the model fits really well with the training data, however it performs slightly less accuracy with the new data. The loss values also are a sign that the model is overfitting.

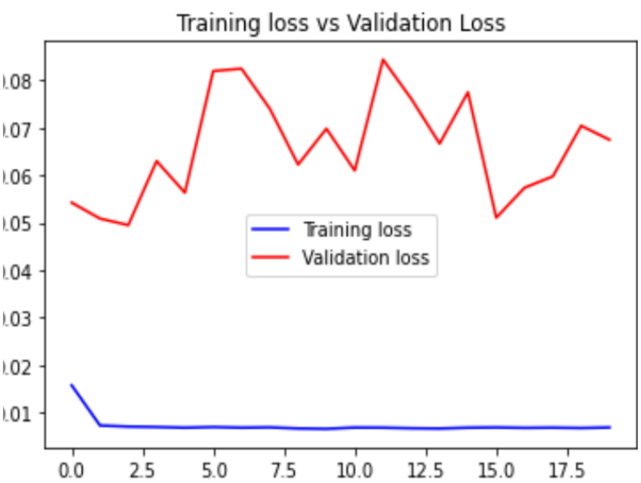

Fig. 12. Training loss vs Validation loss

IV. Results

From Fig. 11 we have the epoch set to 25 and batch size set to 15 when performing regression fit, while setting the dropout to 0.2. In this case, the definition of epoch is the amount of iteration that an entire dataset is passed forward and backward through the neural network regressor model. Batch size indicate the Total number of training examples present in a single batch. Dropout simply discards a portion of neurons during training, with is 20% in this case.

Different parameters can impact the result differently. However, it is not recommended to optimize the parameter to focus only on one dataset as it will yield overfitting, which the model possibly would not work as great on the other models.

A graph is produced with the predicted data for 30 days against the actual bitcoin price as in Fig. 13.

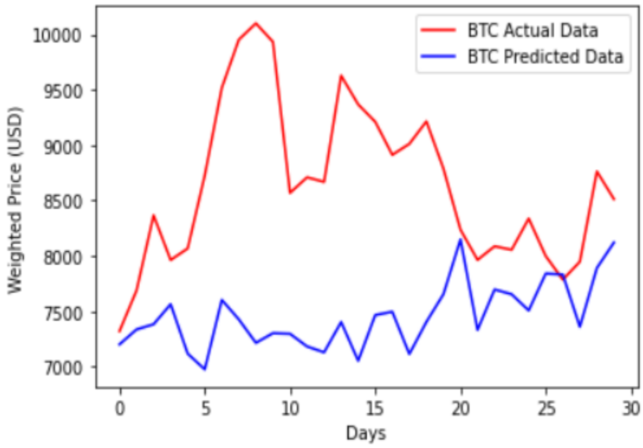

Fig. 13. Bitcoin Predicted Price vs Actual Price

From Fig. 13, it shows that the neural network prediction mode did well on predicting the trend line versus the actual price. However, when we do a comparison on both prices at an instantaneous x-value, there still exist some difference between each other. Furthermore, the predicted values do not react quickly. The trend has a latency in terms of changing values when comparing to the actual data. This could be due to the unpredictability of bitcoin's nature with different real-world uncertain issues that can make a significant impact on it instantly. These could include political issues, global economy effect, top companies' policy and many other factors that are impacting both locally and locally, directly and indirectly.

The graph also proved the statement in training loss and validation loss, where it stated that the model does well when fitting the training data but performs not as good when fitting testing data or new data in general.

Overall, the model succeeded to predict the bitcoin from given historical pricing data. However, there are rooms and many ways to improve the prediction accuracy. A few examples can be: 1. Select different proportion of training and testing size. 2. Use a different training model. 3. Change the parameters when fitting and performing regression, however that could lead to overfitting.

V. Conclusion

In summary, we studied the benefits of recurrent neural network and its use in the real-world application – bitcoin pricing prediction. A neural network model is built to analyze key information on the model. We found that the model predicts the bitcoin price (extension from the result). RNNs and LSTM are great architectures we can use to analyze and predict time-series information not only restricted to bitcoin, but also other time series information like weather, real estate trend, stock market and others[9].

A great extension to the study is to compare how different neural networks predict against time series data, for example vanilla neural network, artificial neural network and others (Fig. 14). There could be a lot more interesting correlation we observe from the models.

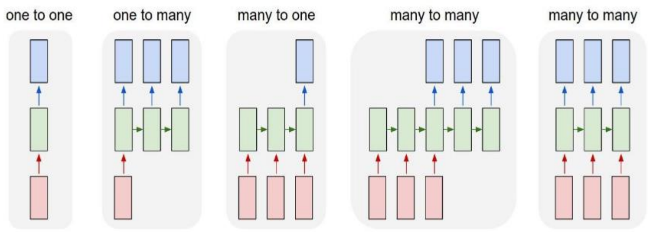

Fig.14. Different types of neural network, image by Andrej Karpath

It is encouraged for anyone who has an interest in neural networks, or general data science field to research and implement recurrence neural networks for any purposes. In terms of application, it will be important to evaluate the performance and accuracy aspect of the network on the object. Comparison between different neural networks on topics are also a great way to help researchers to understand and create a positive impact. Finally, adopting different networks towards other domains and publishing the results are highly respected.

Acknowledgment

I would like to give a huge thank to Dr. Neil D. B. Bruce for teaching and consistently supporting me throughout the semester and the project. Furthermore, thanks to the teaching assistants for guiding me during the term. It is my pleasure to have the opportunity to work with and learn from them. Finally, thank you for your time reading.

References

[1] "Bitcoin - Wikipedia", En.wikipedia.org, 2021. [Online]. Available: https://en.wikipedia.org/wiki/Bitcoin. [Accessed: 10- Dec- 2021].

[2] "Recurrent neural network - Wikipedia", En.wikipedia.org, 2021. [Online]. Available: https://en.wikipedia.org/wiki/Recurrent_neural_network. [Accessed: 10- Dec- 2021].

[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Comput., vol. 9, no. 8, pp. 1735–1780, 1997

[4] M. Zielinski, "Bitcoin Historical Data", *Kaggle.com*, 2021. [Online]. Available: https://www.kaggle.com/mczielinski/bitcoin-historical-data. [Accessed: 10- Dec- 2021].

[5] M. Mayank, "A practical guide to RNN and LSTM in Keras", *Medium*, 2021. [Online]. Available: https://towardsdatascience.com/a-practical-guide-to-nn-and-lstm-in-keras-980f176271bc. [Accessed: 10- Dec- 2021].

[6] "Understanding LSTM Networks -- colah's blog", *Colah.github.io*, 2021. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Accessed: 10- Dec- 2021].

[7] A. Sherstinsky, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network", *Physica D: Nonlinear Phenomena*, vol. 404, p. 132306, 2020. Available: 10.1016/j.physd.2019.132306

[8] R. Mittal, S. Arora, and M. P. S. Bhatia, "AUTOMATED CRYPTOCURRENCIES PRICES PREDICTION USING MACHINE LEARNING," 2018.

[9] J. Brownlee, "A Gentle Introduction to Long Short-Term Memory Networks by the Experts", Machine Learning Mastery, 2021. [Online]. Available: https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/. [Accessed: 10- Dec- 2021].