# Predictive Analysis of Cryptocurrency Price Using Deep Learning

Nilabja Bhattacharya 2018201036
Ajay Jadhav 2018201095
Nitish Dwivedi 2018201068
Ravi Hooda 2018201041

***Abstract*— The decentralization of cryptocurrencies has greatly reduced the level of central control over them, impacting international relations and trade. Further, wide fluctuations in cryptocurrency price indicate an urgent need for an accurate way to forecast this price. This paper proposes a novel method to predict cryptocurrency price by considering various factors such as market cap, volume, circulating supply, and maximum supply based on deep learning techniques such as the recurrent neural network (RNN), Gated Recurrent Unit (GRU), Convolution1D and the long short-term memory (LSTM),which are effective learning models for training data, with the LSTM being better at recognizing longer-term associations. The proposed approach is implemented in Python and validated for benchmark datasets. The results verify the applicability of the proposed approach for the accurate prediction of cryptocurrency price.**

***Index Terms*— Bitcoin, LSTM, GRU, CONV1D, Sentiment Analysis, Timeseries**

## I. INTRODUCTION

Time series forecasting or prediction is a well-known problem. Much research has been done for predicting markets such as the stock market. Cryptocurrencies can be considered a form of virtual currency intended to serve as a medium of exchange and presents an interesting topic since it can be treated as a time series prediction problem. This problem still remains in nascent stages. Consequently, there is high volatility in the market, and this offers opportunities for further research on the prediction of cryptocurrency price.

Moreover, cryptocurrencies such as the Bitcoin are increasingly adopted across the world. Because of the open nature of the cryptocurrency, it operates on a decentralized, peer-to-peer, and trustless system in which all transactions are passed to an open ledger known as the blockchain. Such transparency is unknown in the world of classical financial markets.

This paper proposes a framework for predicting cryptocurrency price using deep learning techniques by considering the nonlinear nature of cryptocurrency price, and the following section discusses the features of deep learning techniques.

## II. DEEP LEARNING TECHNIQUES

### A. Types of machine learning problem

Machine learning falls into two categories: supervised learning and unsupervised learning. Supervised learning consists of modeling datasets with labelled instances, whereas unsupervised learning has no such requirement. In supervised learning, each instance can be represented as a set of attributes and target classes. These attributes are mapped into target classes. Examples of supervised methods include neural networks and support vector machines. In the case of unsupervised learning, similar data instances are grouped into clusters. Examples of unsupervised learning include clustering techniques.

### B. Multi-layer perceptron

The multilayer perceptron (MLP) is a simple feed-forward neural network that is most commonly used in classification tasks. In terms of neural network terminology, examples fed to the model are known as inputs, and predicted values are known as outputs. Each modular subfunction is a layer. A model consists of input and output layers, with layers between these known as hidden layers. Each output of one of these layers is a unit that can be considered analogous to a neuron in the brain. Connections between these units are known as the weight, which is analogous to a synapse in the brain. The weight defines the function of the model since this weight is the parameter that is adjusted when training a model. However, the MLPs effectiveness is limited with the vanishing-gradient problem. Here, as layers and time steps of the network are related to each other through multiplication, derivatives are susceptible to exploding or vanishing gradients. Vanishing gradients are more of a concern as they can become too small for the network to learn, whereas exploding gradients can be limited using regularization. Another limitation of the MLP is that its signals only pass forward in a network in a static nature. As a result, it does not recognize the temporal element of a time series task in an effective manner since its memory can be considered frozen in time. The MLP can be considered to treat all inputs as a bucket of objects with no particular order in terms of time. As a result, the same weight is applied to all incoming data, which is a naive approach. The RNN, also known as a dynamic neural network, addresses some of these limitations.

### C. Recurrent Neural Network (RNN)

The structure of the RNN is similar to that of the MLP, but signals can be both forwards and backwards in an iterative manner. To facilitate this, another layer known as the context layer is added. In addition to passing inputs between layers, the output of each layer is fed to the context layer to be fed into the next layer with the next input. In this context, the state is overwritten at each timestep. This offers the benefit of allowing the network to assign particular weights to events that occur in a series rather than the same weight to all inputs, as with the MLP. This results in a dynamic network. In one sense, the length of the temporal window is the length of the

network memory. It is an appropriate technique for a time series prediction task. While this addresses the temporal issue in a time series task, vanishing gradient can still be an issue. In addition, some studies have found that, while the RNN can handle long-term dependencies, it often fails to learn in practice because of difficulties between gradient descent and long- term dependencies.

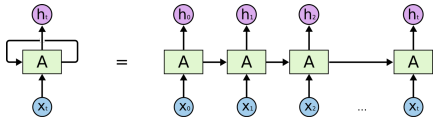


Fig. 1. RNN

### D. Long short-term memory (LSTM)

LSTM units address both these issues [10]. They allow the preservation of weights that are forward and back-propagated through layers. This is in contrast to the RNN, in which the state gets overwritten at each step. LSTM units also allow the network to continue learning over many time steps by maintaining a more constant error. This allows the network to learn long-term dependencies. An LSTM cell contains forget/remember gates that allow the cell to decide what information to block or pass based on information strength and importance. As a result, weak signals can be blocked, preventing the vanishing gradient. LSTM cell states have three dependencies that can be generalized as previous cell states, previous hidden states, and current time steps. These states are accountable for memorizing things, and special gates are used for manipulating this memory. These gates are forget gates, input gates, and output gates. As the name indicates, forget gates remove information that is no longer mandatory for the LSTM. Any addition of new information to the cell state is done using the input gate. The input gate makes use of the tanh function, which gives the output in the form of -1 to +1. The input gate ensures that all redundant information is removed and only the most important information is present. The selection of the most beneficial information from the cell state and its display are the main task of the output gate.

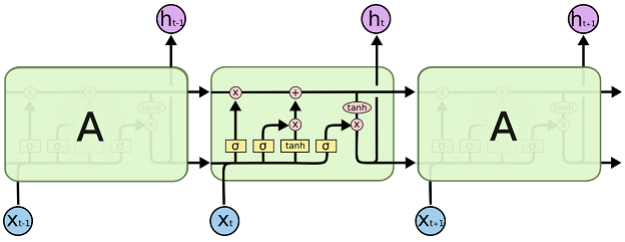


Fig. 2. LSTM

### E. Gated Recurrent Unit (GRU)

The GRU is like a long short-term memory (LSTM) with forget gate but has fewer parameters than LSTM, as it lacks an output gate. GRU's performance on certain tasks of polyphonic music modeling and speech signal modeling was found to be similar to that of LSTM. GRUs have been shown to exhibit even better performance on certain smaller datasets.

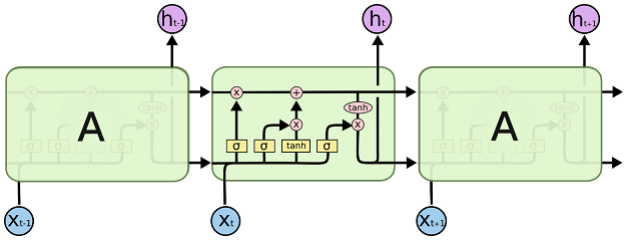


Fig. 3. GRU

### F. Convolutional Neural Network (CNN)

CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually refer to fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks make them prone to overfitting data. Typical ways of regularization includes adding some form of magnitude measurement of weights to the loss function. However, CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme.

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

We have used convolution1D which is just like CNN but for timeseries data.

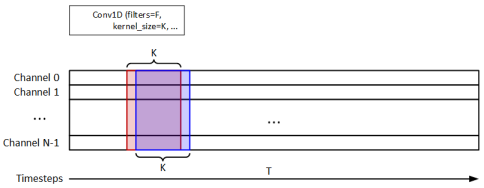


Fig. 4. CONV1D

## III. The Proposed Approach

This section presents the proposed approach for predicting cryptocurrency price using deep learning techniques, as shown in Figs. 1 and 2. The workflow of the proposed approach consists of four units of LSTM input layers for modeling and a sigmoid activation function for controlling the flow of information and memorizing all patterns formed in cryptocurrency data. The section also proposes the use of the adam optimizer to iteratively update network weights for training purposes. The dense layer is passed to make the model more precise. The four LSTM layers used in

the proposed approach help make the model more suitable for learning higher-level representations. LSTM layers return their full output sequences, and the dense layer converts the input sequence into a single vector. This approach uses the square of the correlation coefficient to find the relationship between characteristic fields in the data set. This helps the dominating parameters to derive the values for remaining fields.Then the cryptocurrency price is determined using linear forecast and exponential forecasting. The proposed approach uses the LSTM model to forecast cryptocurrency price.

### A. *The phases of proposed approach are:*

- *Data Analysis Phase:* This phase analyzes data and its parameters to check any redundancy in data values that may affect prediction results. If a dataset contains any irrelevant parameters, then those data values are removed. This phase also analyze data for the possible merging of data for improved model predictability.
- *Data Filtration Phase:* This phase filters data to remove all empty/redundant values.
- *Train-Test Split Phase:* This phase splits data into training and testing data subsets. For example, data are divided into two parts per a ratio of 70% training data and 30% test data.
- *Data-Scaling Phase:* Before data are passed to the model, the data are scaled according to model requirements. In this way, this phase reshapes data to make them more suitable for the model.
- *Model-Building Phase:* The proposed approach is implemented in Python. For any machine learning models, there are two most powerful libraries in Python: Theano and tensorflow. However, these libraries are difficult to use directly for building a model. Therefore, Keras with tensorflow is used as the backend library to make the model more accurate.The Keras sequential model consists of two layers named LSTM and dense layers. These layers process data in depth to analyze all kinds of patterns formed in the dataset to make the model more precise. Then the data are passed to that model for training.
- *Model Learning and Evaluation Phase:* Data are trained using various LSTM units. This consists of four gates: a memory cell, an input gate, an output gate, and a forget gate. These gates are used to let information pass through. They consist of activation layers such as a sigmoid that outputs numbers between 0 and 1. Here 0 means let nothing through, and 1 means let everything through. These gates are used to protect and control the cell state.
- *Prediction Phase:* Prediction is made using the saved model. Input values are passed to the model to give predicted values as the output. Then that output is compared with testing data to calculate accuracy and losses.

## IV. Data Analysis

### A. *Correlation between different fields of bitcoin dataset*

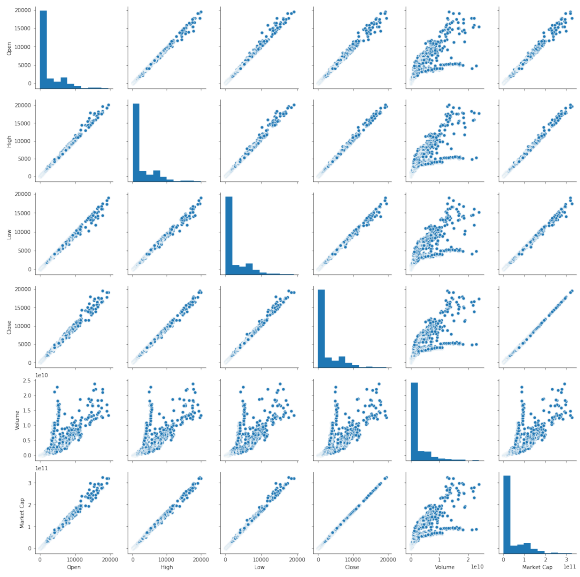

Fig. 5. Correlation between different fields of bitcoin dataset

### B. *Correlation Between Different Fields*

- *Correlation Between Open And Low Price* The value of square of coefficient of 0.99 infers high correlation between Open(x-axis) And Close Price(y-axis). LinregressResult(slope=0.9988,intercept=0.44786, rvalue=0.9999, pvalue=0.0,stderr=8.8848e-07)

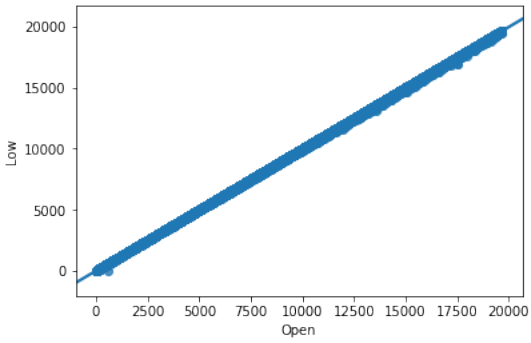


Fig. 6. Correlation Between Open And Low Price[Bit Coin Dataset]

- *Correlation Between Open Price And Volume* The value of square of coefficient of 0.035 infers low correlation between Open Price(x-axis) And Volume(y-axis).Correlation Coefficient value is very low,so it is not a optimal attribute on our model. LinregressResult(slope=0.00032690,intercept=6.6252, rvalue=0.035572, pvalue=0.0,stderr=4.7245e-06)

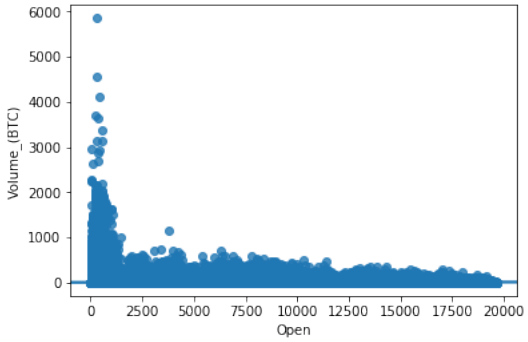


Fig. 7. Correlation Between Price And Volume[Bit Coin Dataset]

- *Correlation Between Open And Close Price* The value of square of coefficient of 0.99 infers high correlation between Open Price(x-axis) And Close Price(y-axis). LinregressResult(slope=0.9999,intercept=0.00676, rvalue=0.9999, pvalue=0.0,stderr=1.0607e-06)

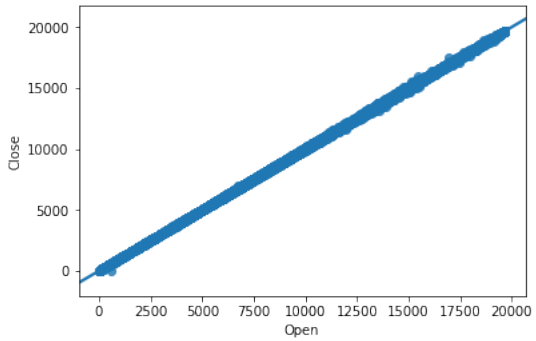


Fig. 8. Correlation Between Open And Close Price[Bit Coin Dataset]

TABLE I
CORRELATION BETWEEN DIFFERENT FIELDS

| Field Value 1 | Field Value 2 | Relationship | Coefficient Value |
|---|---|---|---|
| Open | Close | Linear | 0.99 |
| Open | Volume$(BTC)$ | Linear | 0.035 |
| Open | Low | Linear | 0.99 |

### C. Analysis On Open Price

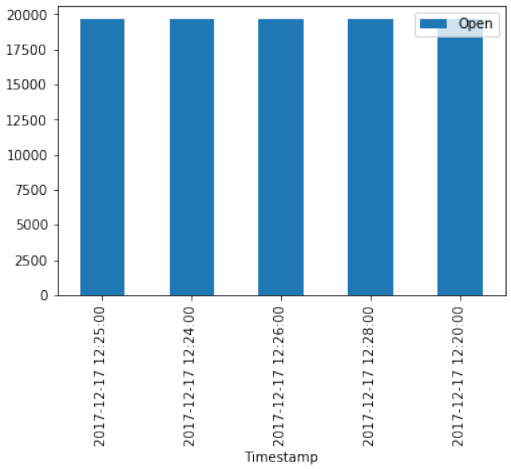


Fig. 9. Top 5 Open Price

Top 5 opening price of cryptocurrency[Bit Coin] has been illustrated in the graph.It is highest on 17th Dec 2017.

### D. Analysis On Close Price

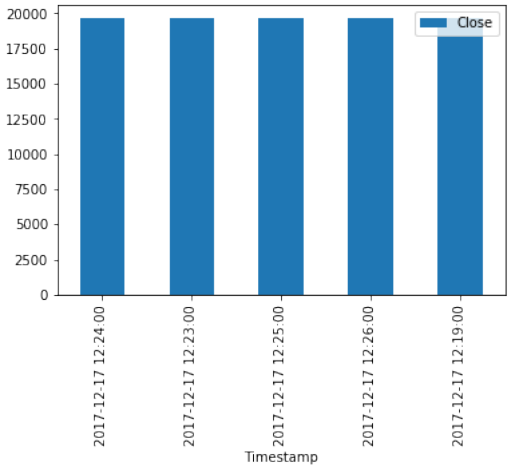


Fig. 10. Top 5 Close Price

Top 5 close price of cryptocurrency[Bitcoin] has been illustrated in the graph.It is highest on 17th Dec 2017.

### E. Analysis On Open Price Variation Over Years

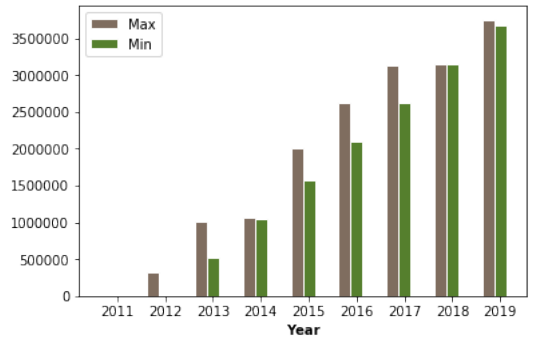


Fig. 11. Minimum and Maximum Open Price

Analysis over maximum and minimum value of opening value of currency[Bit Coin] during each year from 2011-2019 has been illustrated in graph.In the year 2017,currency suffers a largest variation in prices.

### F. Analysis On Volume Over Years

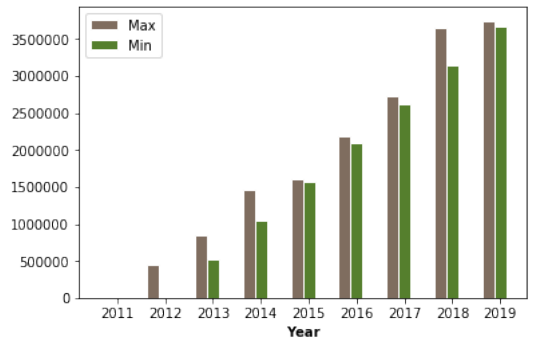


Fig. 12. Minimum and Maximum Volume

Analysis over maximum and minimum value of volume during each year from 2011-2019 has been illustrated in graph.In the year 2018,volume suffers a largest variation in value.

## V. Fluctuation of various cryptocurrency price due to internet activity

It has been seen that internet activity like twitter tweets, reddits activity, google search on cryptocurrency highly affects the price of various cryptocurrency and we represent that using various graph.

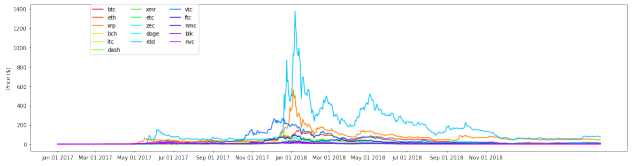

Fig. 13. Fluctuation of various cryptocurrency price from Nov 2017 to Nov 2018

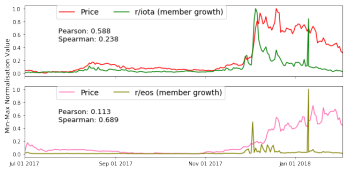


Fig. 14. Fluctuation of various cryptocurrency price based on reddit subscriber count

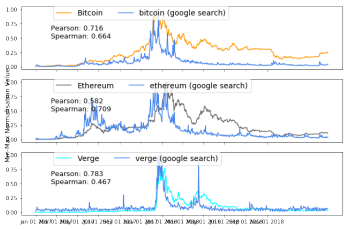


Fig. 15. Fluctuation of various cryptocurrency price based on google search of cryptocurrency

### A. Twitter sentiment analysis to predict bitcoin price

Sentiment analysis has been predominantly used in data science for analysis of customer feedbacks on products and reviews. They are used to understand user ratings on different kinds of products, hospitality services like travel, hotel bookings.

It has also become popular to analyse user tweets - positive, negative or neutral by crawling twitter through APIs.

In this project we have used twitter data, i.e., number of posts on twitter related to bitcoin and then use their polarity to decide how it may affect the bitcoin price.

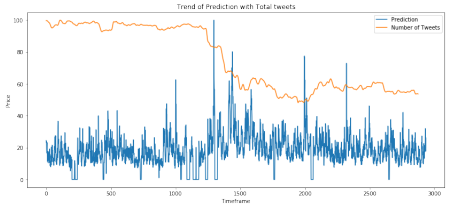


Fig. 16. Variation of bitcoin data with number of tweets

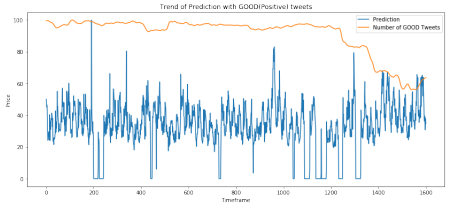


Fig. 17. Variation of bitcoin data with number of positive tweets

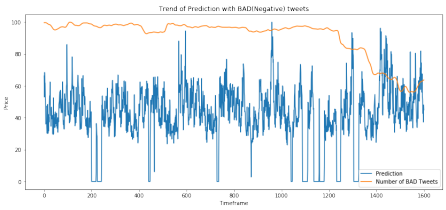


Fig. 18. Variation of bitcoin data with number of negative tweets

## VI. Experimental Setup

The proposed approach is implemented in Python. The implemented model is trained on 5 epochs with 1024 batch size. The proposed approach is executed on google colab. For the validation and ease of debugging, the proposed approach is verified on a single system. Because of the flexibly of the programming language, the same work can be easily extended to the GPU.

TABLE II
Experimental setup

| Parameter | Description |
|---|---|
| GPU | 1xTesla K80 , having 2496 CUDA cores, compute 3.7, 12GB(11.439GB Usable) GDDR5 VRAM |
| CPU | 1xsingle core hyper threaded i.e(1 core, 2 threads) Xeon Processors @2.3Ghz (No Turbo Boost) , 45MB Cache |
| RAM | 12.6 GB Available |
| Disk | 320 GB Available |

### A. Benchmark dataset

The proposed approach was validated using well-known and oldest cryptocurrencies, namely the Bitcoin(BTC) and the Litecoin. The BTC dataset consisted of exchanges for the period January 2012 to March 2019, with minute-to-minute updates of OHLC (Open, High, Low, Close), the volume of BTC and the indicated currency, and weighted Bitcoin prices. The dataset was freely available for use on the Internet. Two Bitcoin datasets corresponded to the U.S. dollar (USD) as USD_Small and USD_Large,and one Bitcoin dataset corresponded to the Japanese yen (JPY). Apart from that I've also extracted dataset of various cryptocurrency from coinmarket and used them to train model and thus predict their prices using LSTM network.

TABLE III
Crytocurrency details

| Cryptocurrency | Data Cols |
|---|---|
| Bitcoin | Open, High, Low, Close, Volume, Weighted Price |
| Litecoin | Open, High, Low, Close, Volume, Market Cap |
| Bitcoin Cash | Open, High, Low, Close, Volume, Market Cap |
| Ethereum | Open, High, Low, Close, Volume, Market Cap |
| Ripple | Open, High, Low, Close, Volume, Market Cap |
| Stellar | Open, High, Low, Close, Volume, Market Cap |
| Cardona | Open, High, Low, Close, Volume, Market Cap |
| EOS | Open, High, Low, Close, Volume, Market Cap |
| Tether | Open, High, Low, Close, Volume, Market Cap |
| Binance | Open, High, Low, Close, Volume, Market Cap |

### B. Performance metrics

The performance of the proposed approach was measured using the most commonly used metrics:

- *Number of epochs:* This is defined as a complete amount of data that must be learned by the machine in a single iteration during the training stage.
- *Amount of losses:* This is defined as the loss of accuracy due to the inefficiency of the prediction model. The reason can include insufficient data and the improper tuning of the prediction model.
- *Correlation coefficient:* The measurement indicating how strong the relationship between two variables. The Person correlation coefficient is the most commonly used method, and its formula for any two relationships x and y is given by:
  r = n $\frac{\sum xy - \sum x \sum y}{\sqrt{n[\sum x^2 - (\sum x)^2][\sum y^2 - (\sum y)^2]}}$
- *Mean absolute scaled error(MASE):* This is a measure of the accuracy of forecasts. Forecast errors for a given period are represented by the numerator, and its value is calculated by subtracting the forecast value($F_t$ ) from the actual value($Y_t$ ) as $e_t = Y_t - F_t$ . For a nonseasonal time series,the denominator is the representation of the mean absolute error of one step using the naive forecast method for the given dataset. Its value is calculated by $F_t = Y_{t-1}$ using the prior period as the new forecast represented by:
  $MASE = \frac{1}{T} \frac{\sum |e_t|}{1/T-1 \sum |Y_t - Y_{t-1}|}$
- *Systematic mean absolute percentage error:* This is the measurement of accuracy based on the percentage error. Here the actual value is represented by At, and Ft represents the forecasted value. The absolute difference between F t and A t is divided by their sum, followed by the further summation of all fitted points divided by the count n of fitted points:
  $SMAPE = 100\%/n \sum \frac{|F_t - Y_t|}{|A_t| - |F_t|/2}$

- *Mean absolute error (MAE):* This measures the difference between two continuous variables. For a given scatter plot with n points, any point i is represented by coordinates (xi, yi), and the MAE is the average vertical distance between each point and the line Y=X. This line is called a one-to-one line:
  $MAE = \sum e/n$
- *Mean square error:* This measures the error between two datasets.This is calculated as the sum of all observations calculated as the difference between the predicted value($P_i$) and the observed value($O_i$) divided by the number of observations(n): $MSE = \sum P_i - O_i/n$

## VII. Results and Comparisons

The model was executed and implemented for benchmark datasets. We have built various model and the best performance was for the one with four LSTM layers of 32, 64, 128, 256 units followed by a dense layer, in this model we have used time series of *close* column to predict *close* with a timestep of 100. Apart from this we have also trained various other models with varying timestep and we have also used various other hidden units i.e. GRU, CONV1D.

### A. Performance of model on bitcoin dataset

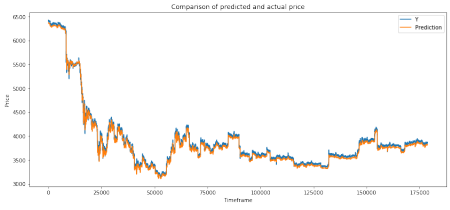


Fig. 19. Performance on bitcoin large dataset

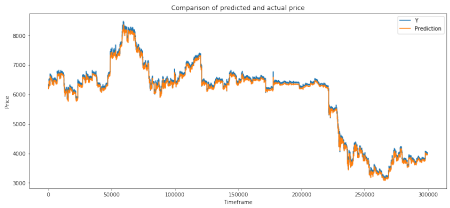


Fig. 20. Performance on bitcoin small dataset


Fig. 21. Performance on bitcoin large dataset[avg open,close]

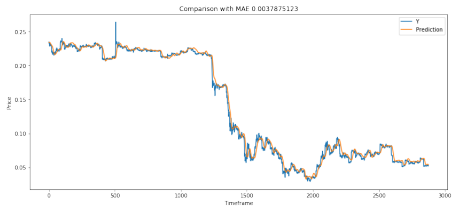


Fig. 22. Performance when model trained by incorporating tweets made by user

TABLE IV
ERRORS ON BITCOIN LARGE AND BITCOIN SMALL DATASET

| Architecture | MSE | MASE | SMAPE | MAE |
|---|---|---|---|---|
| Bitcoin Large | 4.653809e-06 | 9.217731e-05 | 1.062636 | 0.002047 |
| Bitcoin Small | 4.16821e-05 | 0.000194 | 1.96753 | 0.006100 |
| Bitcoin Large[avg open,close] | 2.41e-04 | 0.0732 | 5.1432 | 0.002411 |

### B. Performance comparison of various NN architecture

TABLE V
ERROR COMPARISON USING VARIOUS NN ARCHITECTURE

| Architecture | MSE | MASE | SMAPE | MAE |
|---|---|---|---|---|
| LSTM (32, 64, 128, 256) | 1.64761e-05 | 0.00019 | 1.94857 | 0.00392 |
| GRU (32, 64, 128, 256) | 1.96731e-05 | 0.00019 | 1.95842 | 0.00400 |
| CONV1D ((64,3), (32, 3)) | 2.61811e-05 | 0.00024 | 2.52501 | 0.00497 |

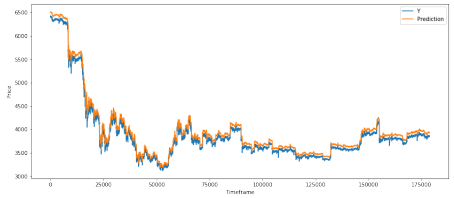


Fig. 23. Actual vs Predicted using LSTM (32, 64, 128, 256)

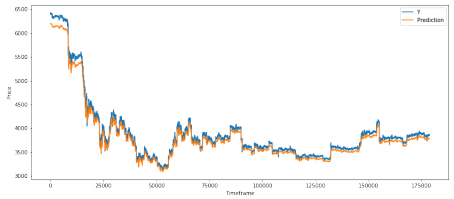


Fig. 24. Actual vs Predicted using GRU (32, 64, 128, 256)

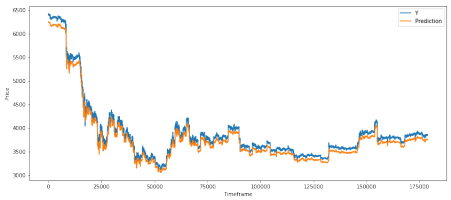

Fig. 25. Actual vs Predicted using CONV1D ((64,3), (32, 3))

### C. Performance on various crytocurrency

TABLE VI
ERROR COMPARISON ON VARIOUS CRYTOCURRENCY

| Cryptocurrency | MAE | SMAPE | MAPE |
|---|---|---|---|
| Bitcoin | 0.01987 | 7.33737 | 0.01064 |
| Litecoin | 0.09476 | 8.39628 | 0.00730 |
| Bitcoin Cash | 0.00671 | 11.91254 | 0.26043 |
| Ethereum | 0.00671 | 11.91254 | 0.26043 |
| Ripple | 0.01552 | 14.75899 | 0.013525 |
| Stellar | 0.01680 | 10.26539 | 0.01185 |
| Cardona | 0.00362 | 6.70209 | 0.14048 |
| EOS | 0.01659 | 7.98707 | 0.06755 |
| Tether | 0.014260 | 2.17750 | 0.01348 |
| Binance | 0.09476 | 11.97607 | 0.12174 |

### D. Error comparison on increasing timestep on LSTM (50, 50, 50, 50)

TABLE VII
ERROR COMPARISON ON VARIOUS CRYTOCURRENCY ON 25000000 DATA

| Timestep | MAE | SMAPE | MAPE | MSE |
|---|---|---|---|---|
| 50 | 0.00561 | 2.97093 | 0.00106 | 3.17826e-05 |
| 100 | 0.00215 | 1.15739 | 0.00041 | 4.76673e-06 |
| 150 | 0.00207 | 1.10784 | 0.00039 | 4.45528e-06 |
| 200 | 0.00126 | 0.67959 | 0.00024 | 1.67421e-06 |

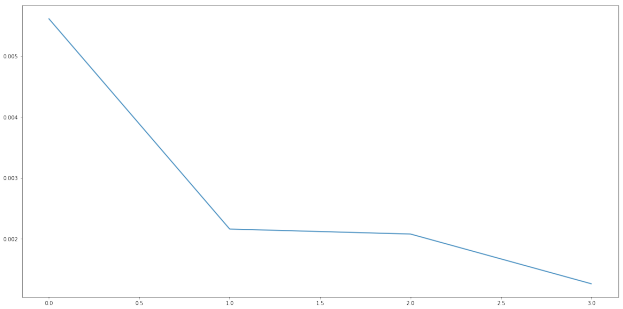

Fig. 26. MAE with increasing timestep

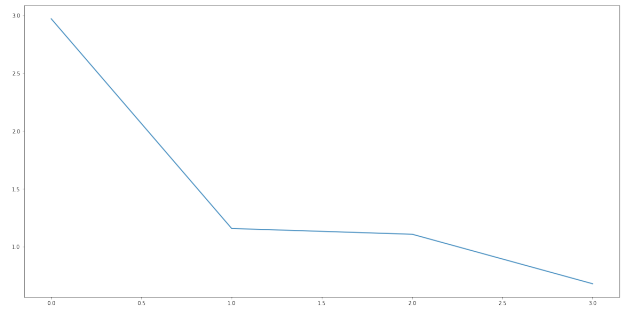

Fig. 27. SMAPE with increasing timestep

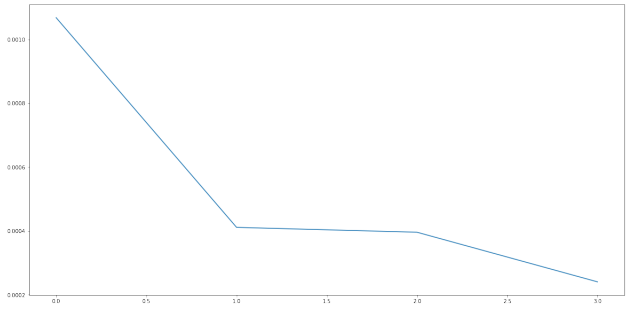

Fig. 28. MASE with increasing timestep

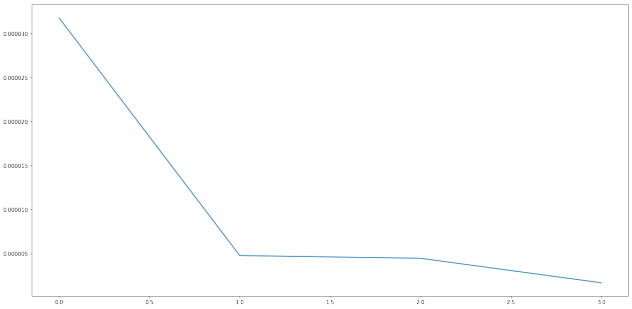

Fig. 29. MSE with increasing timestep

### E. Error comparison with varying hidden units of LSTM

TABLE VIII
ERROR COMPARISON ON VARIOUS CRYPTOCURRENCY ON 25000000 DATA

| Hidden Units | MAE | SMAPE | MAPE | MSE |
|---|---|---|---|---|
| 30 | 0.00588 | 2.75189 | 0.000184 | 3.71176e-05 |
| 60 | 0.00935 | 4.72756 | 0.00029 | 8.81446e-05 |
| 90 | 0.00259 | 1.56483 | 8.154593e-05 | 1.03441 |
| 120 | 0.00216 | 0.95151 | 6.81915e-05 | 5.77573e-06 |
| 150 | 0.00083 | 0.45784 | 2.61942e-05 | 1.10226e-06 |

## VIII. CONCLUSIONS

The decentralization of cryptocurrencieshas sharply weakened central control. Further, wide fluctuations in cryptocur-

rency price indicate an urgent need for methods to accurately forecast cryptocurrency price. This paper proposes a novel method to predict cryptocurrency price by considering various factors such as market cap, volume, circulating supply, and maximum supply based on deep learning techniques such as the RNN, CONV1D, GRU and the LSTM. The proposed approach is implemented in Python and validated for benchmark data sets. The results verify the applicability of the proposed approach for the accurate prediction of cryptocurrency price. We have also incorporated how twitter data can lead to fluctuation of bitcoin price. Future research should extend the proposed approach by considering additional parameters such as the political environment, human relations, and regulations, which vary across countries.

## IX. Contribution by candidates

- Nilabja Bhattacharya (2018201036)
  - Performance of LSTM on primary bitcoin dataset
  - Performance comparison on using LSTM, GRU and CONV1D
  - Performance comparison on increasing timestep
  - Performance of LSTM on various cryptocurrency
  - Extraction of various cryptocurrency data from coinmarket
- Ajay Jadhav (2018201095)
  - Analysis of google trends (Bitcoin search vs bitcoin price)
  - Analysis of cryptocurrency price with varying subscribers of reddit cryptocurrency pages
- Nitish Dwivedi (2018201068)
  - Cryptocurrency price variation with twitter sentiment analysis
  - Performance comparison on increasing hidden units
- Ravi Hooda (2018201041)
  - Data analysis of bitcoin dataset
  - Performance of bitcoin price prediction using average, open and close price of bitcoin datasett

Appendixes should appear before the acknowledgment.

## X. ACKNOWLEDGMENT

We are very thankful to the TA for all support and guidance required for the completion of the projects. Apart from that very thankful to Google Colab that let us complete the project with the help of GPU.

## References

[1] www.kaggle.com
[2] ww.coinmarket.com
[3] Predictive Analysis of Cryptocurrency Price Using Deep Learning
[4] Bitcoin Response to Twitter Sentiments
[5] Stock Prediction Using Twitter Sentiment Analysis