# Bitcoin Price Prediction Using Deep Learning

Anas Saifi

March 13, 2022

# BITCOIN PRICE PREDICTION USING DEEP LEARNING

Anas Saifi

Master of Science (Computer Science)
University of Mumbai
Thakur College of Science and Commerce Kandivali (East), Mumbai - 400067, India.

***Abstract :*** Bitcoin has recently received a lot of attention from the media and the public due to its recent price surge and crash. Correspondingly, many researchers have investigated various factors that affect the Bitcoin price and the patterns behind its fluctuations, in particular, using various machine learning methods. In this paper, we study and compare various state-of-the-art deep learning methods such as a deep neural network (DNN), a long short-term memory (LSTM) model, Gated recurrent unit (GRU) model, a fast implementation of GRU backed by cuDNN, and Recurrent Neural Network(RNN) for Bitcoin price prediction. Experimental results showed that although LSTM-based prediction models slightly outperformed the other prediction models for Bitcoin price prediction (regression), DNN-based models performed the best for price ups and downs prediction (classification). Overall, the performances of the proposed deep learning-based prediction models were comparable**.**

***IndexTerms*** **- Bitcoin Price Prediction, LSTM, GRU, CuDNNGRU, Simple RNN.**

## I. INTRODUCTION

Bitcoin is a decentralized digital currency (i.e a Cryptocurrency) without a central bank or administrator. It can be sent from user to user on the Bitcoin network without the need of intermediaries[1]. Bitcoin transactions are verified by network nodes through cryptography and recorded in a public distributed ledger called a Blockchain. Bitcoin offers an efficient means of transferring money over the internet since its controlled by a decentralized network with a transparent set of rules, thus presenting an alternative to central bank-controlled fiat money. This is the main reason behind the success and popularity of Bitcoin. Just like any other currency, the price of Bitcoin changes every day. The only difference is that the price of Bitcoin changes on a much greater scale than local currency (current price USD 44k) . Therefore it is important for investors to not invest more money than they can afford to lose. This problem makes it important to know how much the price is going to vary in future. Thus the problem of predicting Bitcoin price is an important problem.

## II. LITERATURE REVIEW

In [2], Artificial Neural Network (ANN) is used for predicting the next day price of bitcoin. They used four ANN algorithms namely Neuro Evolution of Augmenting Topologies (NEAT), Genetic Algorithm Neural Network (GANN), Genetic Algorithm Backpropagation Neural Network (GABPNN) and Backpropagation Neural Network (BPNN). These methods are compared based on their algorithmic complexity and accuracy of the results. In [3], bitcoin price prediction is modelled as a binomial classification problem using machine learning algorithms like generalized linear models and random forests.

[4] focuses on optimal features affecting bitcoin prices and proposed price prediction approach using different machine learning algorithms like Bayesian Regression and Random forest. [5] uses Bitcoin Price Index as data source and compares the ARIMA model with various deep learning models like Bayesian optimised recurrent neural network (RNN) and a Long Short Term Memory (LSTM) network on the basis of accuracy and Root Mean Square Error (RMSE) for bitcoin price prediction. [6] compares linear regression model with polynomial regression models like Support Vector regression and KNN regression and on the basis of RMSE, it concludes that Support Vector models have been outperformed by KNN

## III. METHODOLOGY

### 3.1) Preprocessing

The dataset is loaded and split for training and testing. We test/predict the bitcoin price value for 50 days and use the remaining data for training purposes. We also normalize the prices to values between 0 and 1 using the scikit learn MinMaxScaler

### 3.2) Simple Recurrent Neural Network(RNN)

We first use the Simple RNN to try and predict the bitcoin price for 50 days. A recurrent neural network has connections between nodes in the form of a directed graph along a temporal sequence[7]. This allows capturing the temporal information from the time series data.

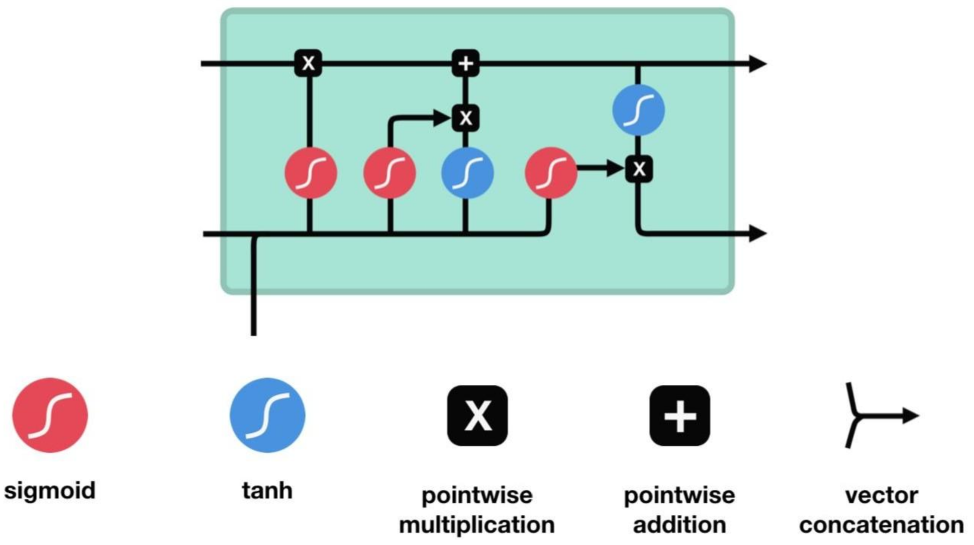
Assuming $H_t$ is the RNN state at time $t$ and $H_{t-1}$ is the RNN state at time $t-1$, and the input to the RNN at instant $t$ is $X_t$, then the RNN can be formulated in the following way:

$$H_t = \sigma_h( W_{hh} H_{t-1} + W_{xh} X_t )$$

Where $W_{hh}$ and $W_{xh}$ are the weights to learn. Learning is done using backpropagation algorithm. Here (and in the following sections) the symbols $\sigma_h$ , $\sigma_g$ respectively are the tanh and sigmoid activation functions. We train the RNN using our training data and make predictions. The results are discussed in later sections.

## 3.3) Long Term Short Memory Network(LSTM)

LSTM is a modified version of the simple RNN with feedback connections and different gates. The simple RNN has many issues like exploding signal, vanishing gradient problem, etc. and the LSTM tried to solve those issues[8]. LSTM has a forget gate, an input gate, an output gate and a memory cell. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.



The LSTM Cell and it's operations

Assuming $f_t$ , $i_t$ ,$o_t$ , $c_t$ , $h_t$ represent the forget gate, input gate, output gate, memory cell and the LSTM state at time $t$, then the LSTM can be formulated as[9]:

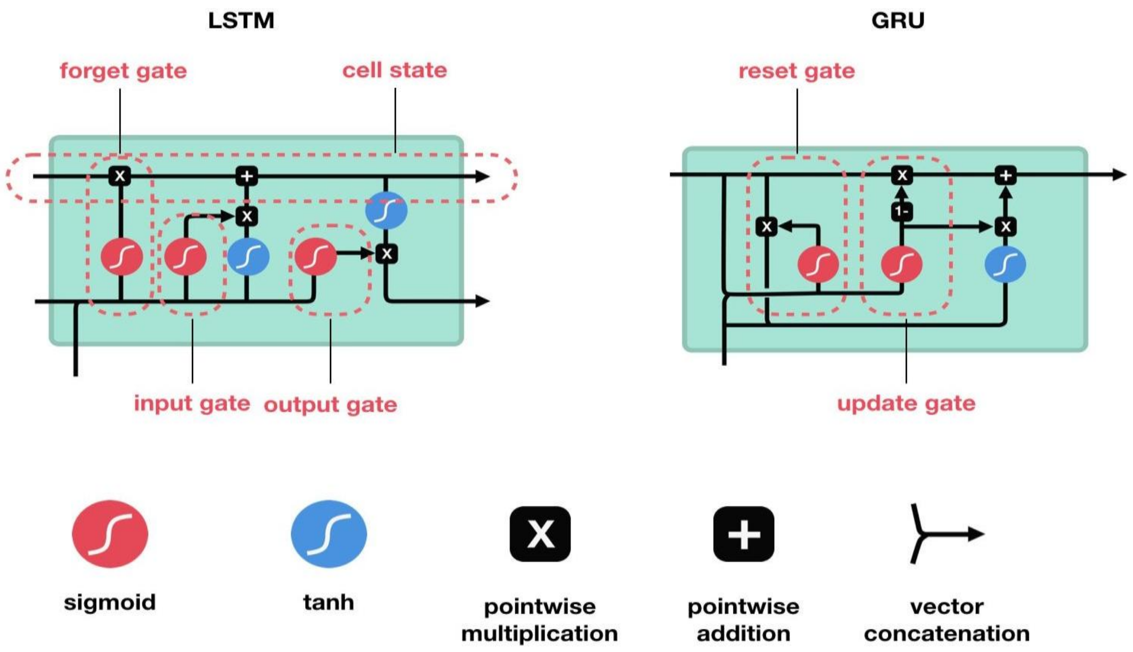
$$\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}$$

Where the terms with $W$ are the different weights that are learnt during the training time. The results obtained using LSTM for bitcoin price prediction are discussed in later sections.

## 3.4) Gated Recurrent Unit Network (GRU) and CuDNNGRU

The GRU is like a long short-term memory (LSTM) with forget gate[9] but has fewer parameters than LSTM, as it lacks an output gate[10]. The gates in a GRU are called the update Gate and the reset gate. Since the GRU has fewer parameters than an LSTM but has the similar architecture, it can work better incase of smaller datasets[11][12]. Thus it works better for our dataset consisting of bitcoin prices. A comparison of the LSTM and GRU structure is highlighted in the below figure:

**FIG**: LSTM and GRU structural differences



Assuming $z_t$ , $r_t$ , $h_t$ are the update gate, reset gate and current state respectively, then GRU is formulated as (again the weights are learned during training) :

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z)$$
$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r)$$
$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \sigma_h(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h)$$

The CuDNNGRU is a slightly different and fast GRU implementation backed by CuDNN. It can only be run on GPU, with the TensorFlow backend

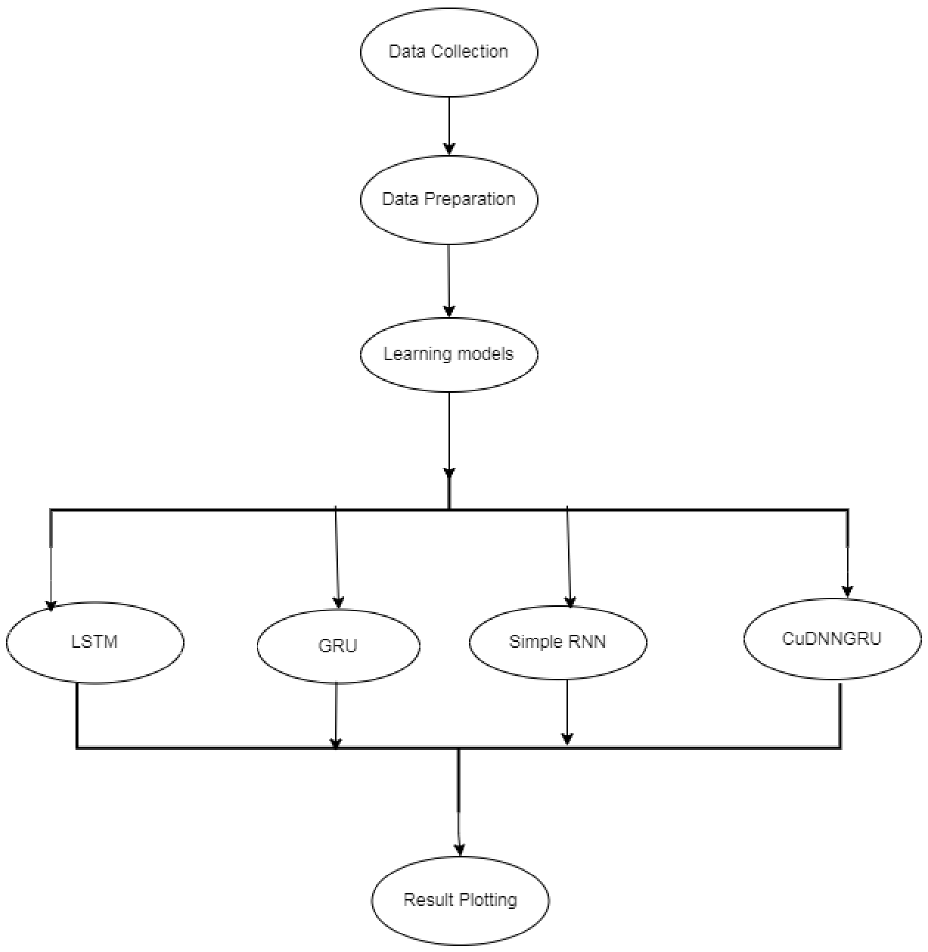
**3.5) Data and Sources of Data**

This is a public dataset downloaded from Kaggle. The original Owner scrapped the data from Bitcoin charts using one minute chart querying and collection of subsequent tabular data. Source. These are CSV files for select bitcoin exchanges for the time period of January 2012 to August 2019, with minute to minute updates of OHLC (Open, High, Low, Close), Volume in BTC and indicated currency, and weighted bitcoin price. Timestamps are in Unix time. Timestamps without any trades or activity have their data fields filled with NaNs. If a timestamp is missing, or if there are jumps, this may be because the exchange (or its API) was down, the exchange (or its API) did not exist, or some other unforeseen technical error in data reporting or gathering.

## IV. Working

The model weights and optimization is highly depended on the initialization point of the weights, therefore randomly initializing the weights leads to some variations in the results each time. Therefore we repeat the experiment multiple times (10 cross-validations) and report the average error obtained for each model. We use the Mean Square Error (MSE) as a metric, it is calculated as the mean squared distance between the actual ( $Y$ ) and the predicted price (Y) :

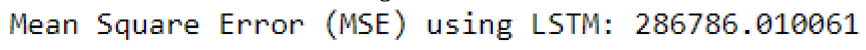
$$MSE = \frac{1}{50}\sum_{1}^{50}(\widehat{Y} - Y)^2$$



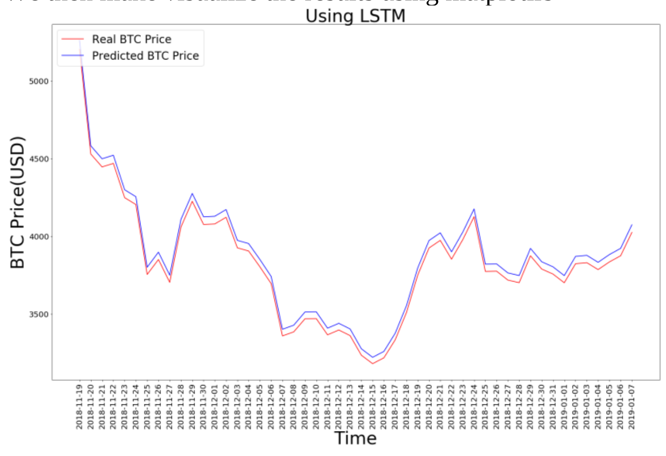
For predicting bitcoin prices I have created a python notebook

- First, we import the common python libraries for all models, I have use numpy, pandas, tensorflow and matplotlib
- We then define the path of the dataset in our notebook.
- We then split the data set based on how for how many days we want to predict.
- We then do some preprocessing, like normalizing the values, reshaping, etc. The data is a time series data, so the output to every instance is the next instance.

**Part 1: Using Long short-term memory (LSTM) network**

- We Import the keras libraries and packages
- We then initialise sequential model from keras and then create LSTM layer and output layer
- We then compile using Adam optimizer and MSE loss
- We then fit the LSTM to the training set
- 

```
Mean Square Error (MSE) using LSTM: 286786.010061
```

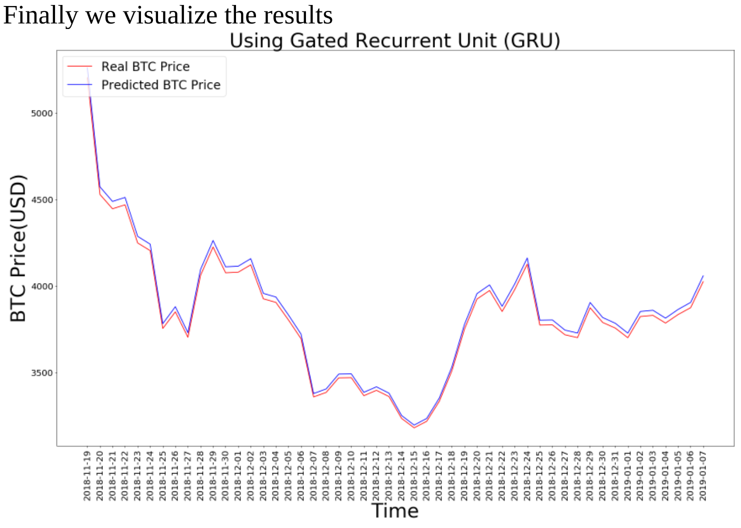
- We then make visualize the results using matplotlib
- 

**Part 2: Using Gated Recurrent Units (GRU)**

- We initialize the sequential model from keras , then we add the GRU Layer and output layer, compile GRU using Adam optimizer and MSE Loss and then we fit GRU to the training set.
- Then we check the predictions from trained GRU network

```
Mean Square Error (MSE) using GRU: 283144.297217
```

-
- Finally we visualize the results



-

**Part 3: Using a different (and faster) GRU implementation backed by CuDNN**

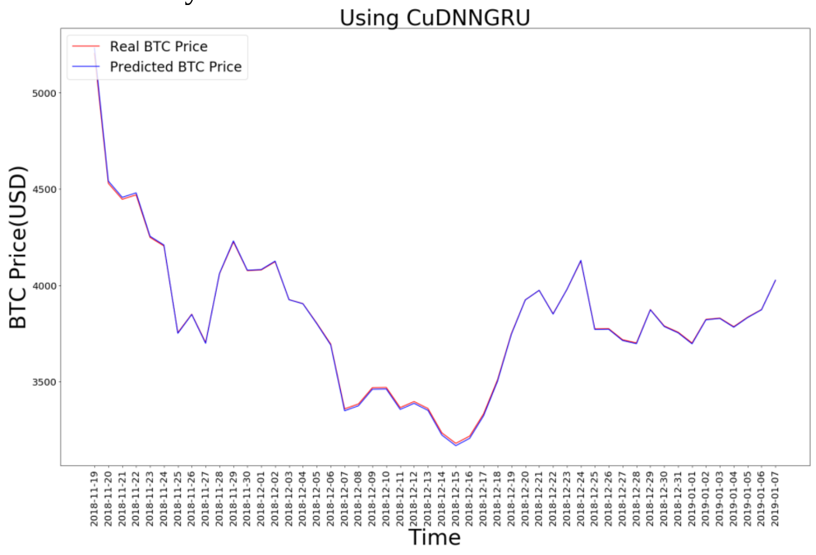
- We initialize the sequential model from keras , then we add a hidden Layer and output layer, compile CuDNN using Adam optimizer and MSE Loss and then we train them

```
Epoch 46/50
95/95 [==============================] - 0s 2ms/step - loss: 1.5438e-04
Epoch 47/50
95/95 [==============================] - 0s 2ms/step - loss: 1.4876e-04
Epoch 48/50
95/95 [==============================] - 0s 2ms/step - loss: 1.5294e-04
Epoch 49/50
95/95 [==============================] - 0s 2ms/step - loss: 1.4824e-04
Epoch 50/50
95/95 [==============================] - 0s 2ms/step - loss: 1.5206e-04
```

- `<keras.callbacks.History at 0x2db6e248>`
- Then we check the predictions from trained CuDNNGRU network

```
Mean Square Error (MSE) using CuDNNGRU: 281242.051223
```

-
- We then finally visualize the results:

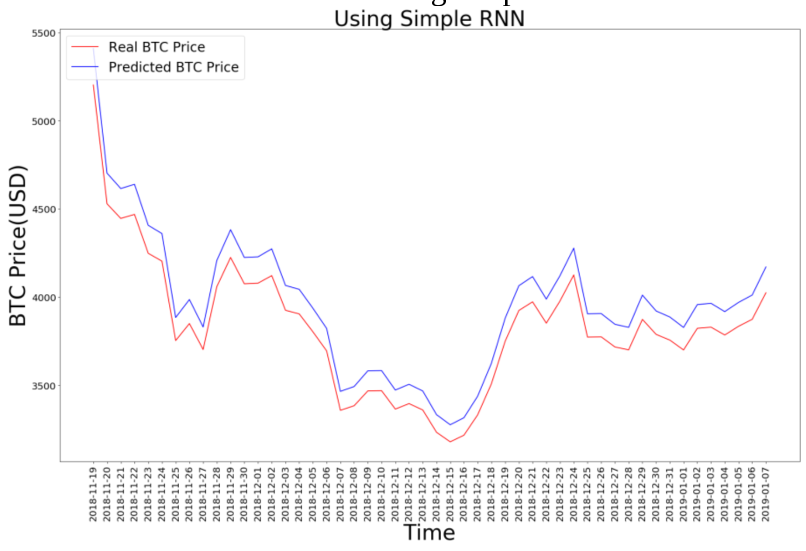


-

**Part 4: Using Simple RNN**

- We first initialize RNN, add simple RNN and the output layer, compile them using Adam optimizer and MSE Loss and then we train them

```
Epoch 45/50
95/95 [==============================] - 0s 1ms/step - loss: 2.2100e-04
Epoch 46/50
95/95 [==============================] - 0s 1ms/step - loss: 2.1975e-04
Epoch 47/50
95/95 [==============================] - 0s 1ms/step - loss: 2.1805e-04
Epoch 48/50
95/95 [==============================] - 0s 1ms/step - loss: 2.2029e-04
Epoch 49/50
95/95 [==============================] - 0s 1ms/step - loss: 2.1620e-04
Epoch 50/50
95/95 [==============================] - 0s 1ms/step - loss: 2.1704e-04
```

- `<keras.callbacks.History at 0x3867ce08>`
- Then we check the predictions from trained RNN network
- `Mean Square Error (MSE) using RNN: 315254.959201`
- We then visualize the results using matplotlib
- 

**V. Conclusion**

| Method / Model Used | Mean Square Error ( $USD^2$ ) |
|---|---|
| Simple RNN | 315254.959201 |
| LSTM | 286786.010061 |
| GRU | 283144.297217 |
| CuDNN GRU | 281242.051223 |

TABLE: Comparing the Mean Square Error averaged over 10 validations

We see that the CuDNN GRU model produces the best quantitative results in terms of the average MSE for prediction of bitcoin prices for 50 days. This is also consistent with our initial assumptions because simple RNN has issues like vanishing gradients, LSTM is heavier on parameters thus it might overfit the training set, and GRU is just a different implementation of CuDNN GRU (the later being a lighter and faster version performs better on this dataset).

We compare the plots obtained by the real and predicted bitcoin prices. The curves are plotted using the matplotlib library and the scales (for X and Y axes) have been suitably chosen so that it highlights the difference. We compare Simple RNN (worst result quantitatively) and the output from the CuDNN GRU (best in terms of MSE) to see if both are qualitatively consistent. The X axis represents time in days and the Y axis is the price of Bitcoin in USD on that day

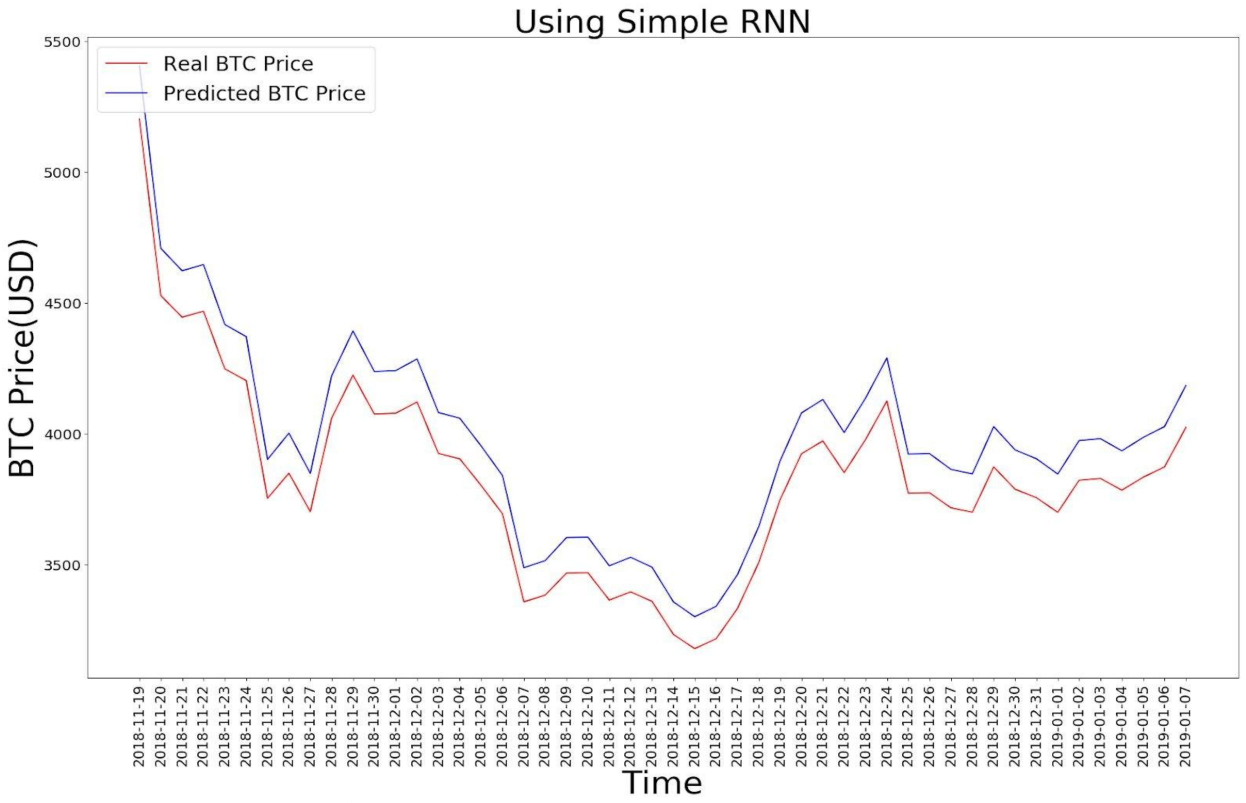


**FIG**: Plot of Bitcoin real and Predicted price by RNN

We see that using simple RNN the predicted price is off from the real price by some amount. Now we see the plot of the prediction by CuDNN GRU

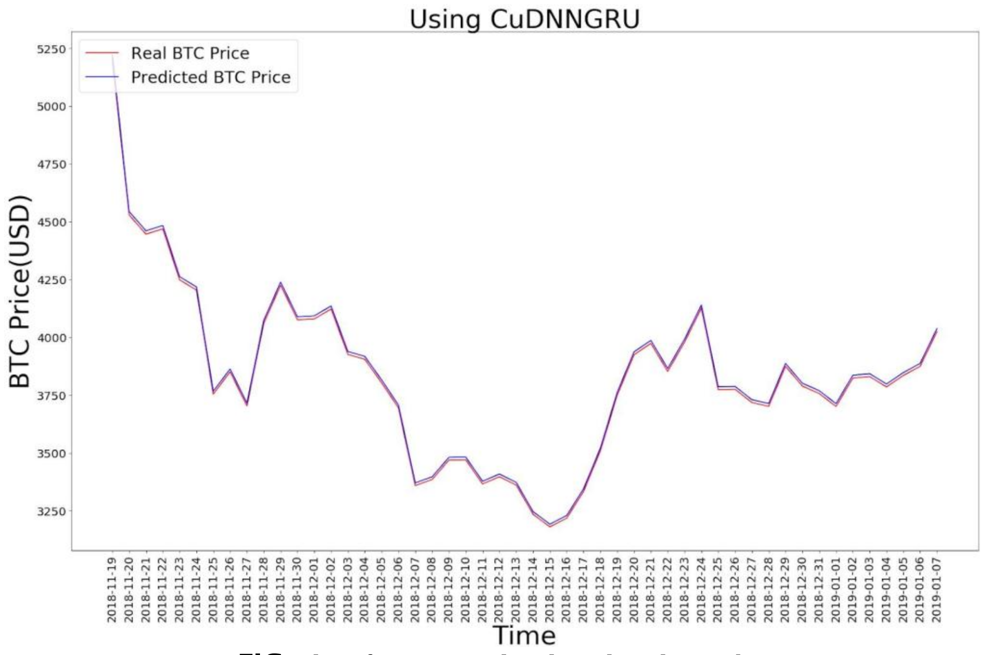


**FIG**: Plot of Bitcoin real and Predicted price by CuDNN GRU

Thus we see that, the prediction by CuDNN GRU is almost overlapping with the real price of bitcoin during the test time interval. The difference between the real and predicted is very small as compared to that of the simple RNN. Although due to the chosen scale it appears almost the same, but there’s still some difference between the actual and the predicted value (which is captured in the MSE loss). One more observation we can conclude is the test set follows similar distribution as that of the training set which might have influenced the reasonably good quality results by all 4 models, but during real world production there is some fluctuations (and other parameters involved) and thus this method might not work so well for large intervals of predictions. But the idea is promising and in future might lead to effective techniques that can accurately predict the future price of Bitcoin (or any other cryptocurrency) by just using the past information.

## REFERENCES

**[1]** "Statement of Jennifer Shasky Calvery, Director Financial Crimes Enforcement Network United States Department of the Treasury Before the United States Senate Committee on Banking, Housing, and Urban Affairs Subcommittee on National Security and International Trade and Finance Subcommittee on Economic Policy" (PDF). fincen.gov. Financial Crimes Enforcement Network. 19 November 2013. Archived (PDF) from the original on 9 October 2016. Retrieved 1 June 2014.

**[2]** Radityo, Arief, Qorib Munajat, and Indra Budi. "Prediction of Bitcoin exchange rate to American dollar using artificial neural network methods." In 2017 International Conference on Advanced Computer Science and Information Systems (ICACSIS), pp. 433-438. IEEE, 2017.

**[3]** Madan, Isaac, Shaurya Saluja, and Aojia Zhao. "Automated bitcoin trading via machine learning algorithms." URL: http://cs229. stanford. edu/proj2014/Isaac 20Madan 20 (2015).

**[4]** Velankar, Siddhi, Sakshi Valecha, and Shreya Maji. "Bitcoin price prediction using machine learning." In 2018 20th International Conference on Advanced Communication Technology (ICACT), pp. 144-147. IEEE, 2018.

**[5]** McNally, Sean, Jason Roche, and Simon Caton. "Predicting the price of Bitcoin using Machine Learning." In 2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), pp. 339-343. IEEE, 2018

**[6]** Singh, Hrishikesh, and Parul Agarwal. "Empirical Analysis of Bitcoin Market Volatility Using Supervised Learning Approach." In 2018 Eleventh International Conference on Contemporary Computing (IC3), pp. 1-5. IEEE, 2018

**[7]** Dupond, Samuel (2019). "A thorough review on the current advance of neural network structures". Annual Reviews in Control. 14: 200–230.

**[8]** Sepp Hochreiter; Jürgen Schmidhuber (1997). "Long short-term memory". Neural Computation. 9 (8): 1735–1780. doi:10.1162/neco.1997.9.8.1735. PMID 9377276. S2CID 1915014.

**[9]** Felix Gers; Jürgen Schmidhuber; Fred Cummins (1999). "Learning to Forget: Continual Prediction with LSTM". Proc. ICANN'99, IEE, London. 1999: 850–855. doi:10.1049/cp:19991218. ISBN 0-85296-721-7.

**[10]** Recurrent Neural Network Tutorial, Part 4 – Implementing a GRU/LSTM RNN with Python and Theano – WildML". Wildml.com. 2015-10-27. Archived from the original on 2021-11-10. Retrieved May 18, 2016.

**[11]** Chung, Junyoung; Gulcehre, Caglar; Cho, KyungHyun; Bengio, Yoshua (2014). "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". arXiv:1412.3555 [cs.NE].

**[12]** Gruber, N.; Jockisch, A. (2020), "Are GRU cells more specific and LSTM cells more sensitive in motive classification of text?", Frontiers in Artificial Intelligence, 3: 40, doi:10.3389/frai.2020.00040, PMC 7861254, PMID 33733157, S2CID 220252321