

Recurrent Networks for Bitcoin price prediction



Rakesh Pavan [Email: rp.171it154@nitk.edu.in | Webpage: <https://rakeshpavan333.github.io/>]

Bachelor of Technology (B.Tech) in Information Technology (Expected 2021)

National Institute of Technology Karnataka (NITK), Surathkal, India

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. 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 10K) . 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.

Problem Statement:

We need to predict the future price of Bitcoin based on only the past and current price information in hand.

Proposed Solution:

We explore and compare the use of deep recurrent networks (namely RNN, LSTM, GRU and CuDNNGRU) to predict the future Bitcoin price. The models are trained on past bitcoin price data available to us. We use recurrent networks because normal neural networks or other machine learning techniques fail to capture the temporal information in the price curve (which is like a time series).

Dataset: Bitcoin Historical Data

A. Contents:

The dataset contains 2 CSV files:




- i. *coinbaseUSD_1-min_data_2014-12-01_to_2019-01-09.csv*
- ii. *bitstampUSD_1-min_data_2012-01-01_to_2019-08-12.csv*

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.

B. Acknowledgements:

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](#) | [Original Owner](#) >

Usage Information	License	CC BY-SA 4.0 
	Visibility	 Public
Provenance	Sources	https://bitcoincharts.com/charts
	Collection methodology	One minute chart querying and collection of subsequent tabular data
Maintainers	Dataset owner	 Zielak
Updates	Expected update frequency	Quarterly
	Last updated	2019-08-15
	Date created	2017-06-02
	Current version	Version 17

Methodology

A. 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](#).

B. 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. 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 σ_h , σ_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.

C. Long Short Term 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. 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.

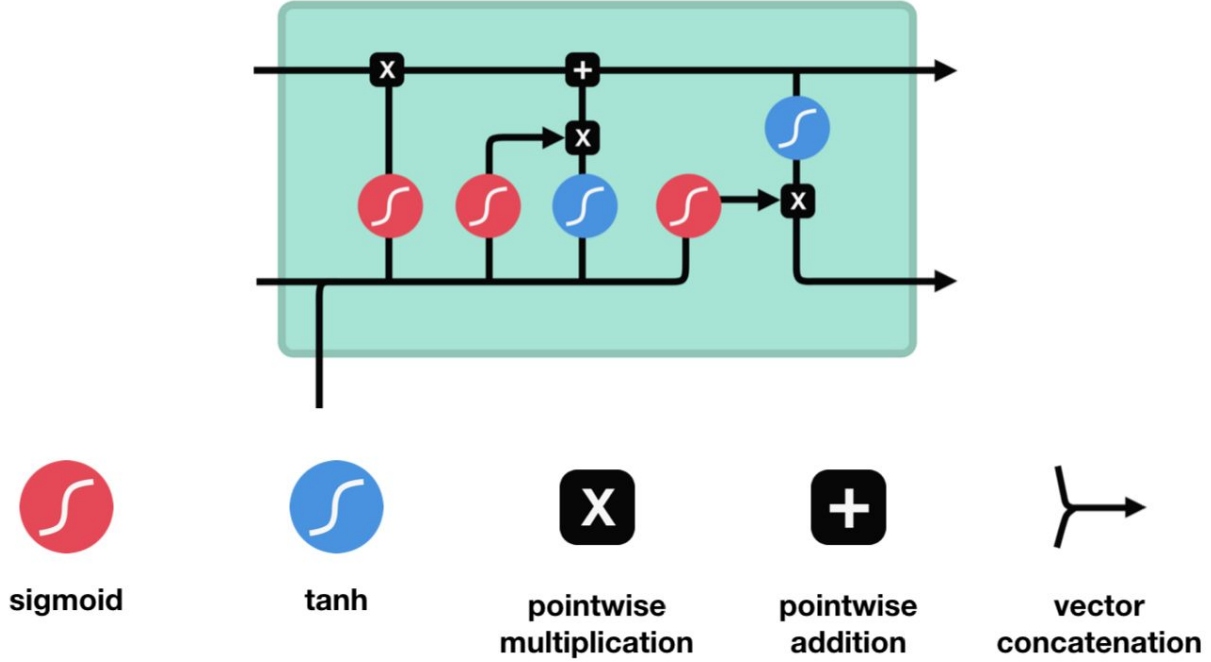


FIG: The LSTM cell and its operations

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:

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

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.

C. Gated Recurrent Unit Network (GRU) and CuDNNGRU

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. 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. 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:

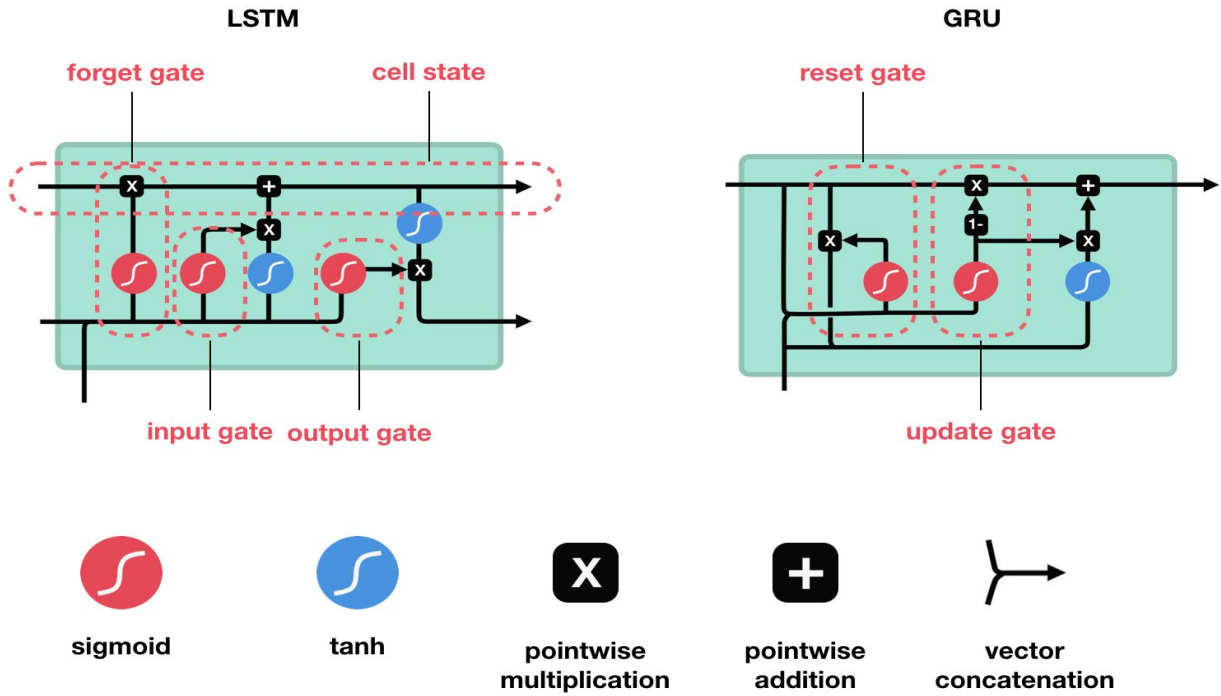


FIG: LSTM and GRU structural differences

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.

D. Implementation

The code and the output can be viewed by opening the file '[RESULT.html](#)' in any browser. We make use of standard library functions from open source. We use [numpy](#) and [pandas](#) library for handling data and numerical operations. We make use of [matplotlib](#) for plotting. [Keras](#) library (with [Tensorflow](#) backend) for implementing the model architectures, training, and predicting. The [scikit-learn](#) is used during preprocessing. Training is done on [Kaggle platform](#) (Nvidia Tesla K80 GPU support with 12 GB RAM).

Results and Discussion

A. Quantitative Analysis

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 (\hat{Y}) and the predicted price (Y) :

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

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).

A. Qualitative Analysis

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 (other plots available in '[RESULT.html](#)' file). The X axis represents time in days and the Y axis is the price of Bitcoin in USD on that day.

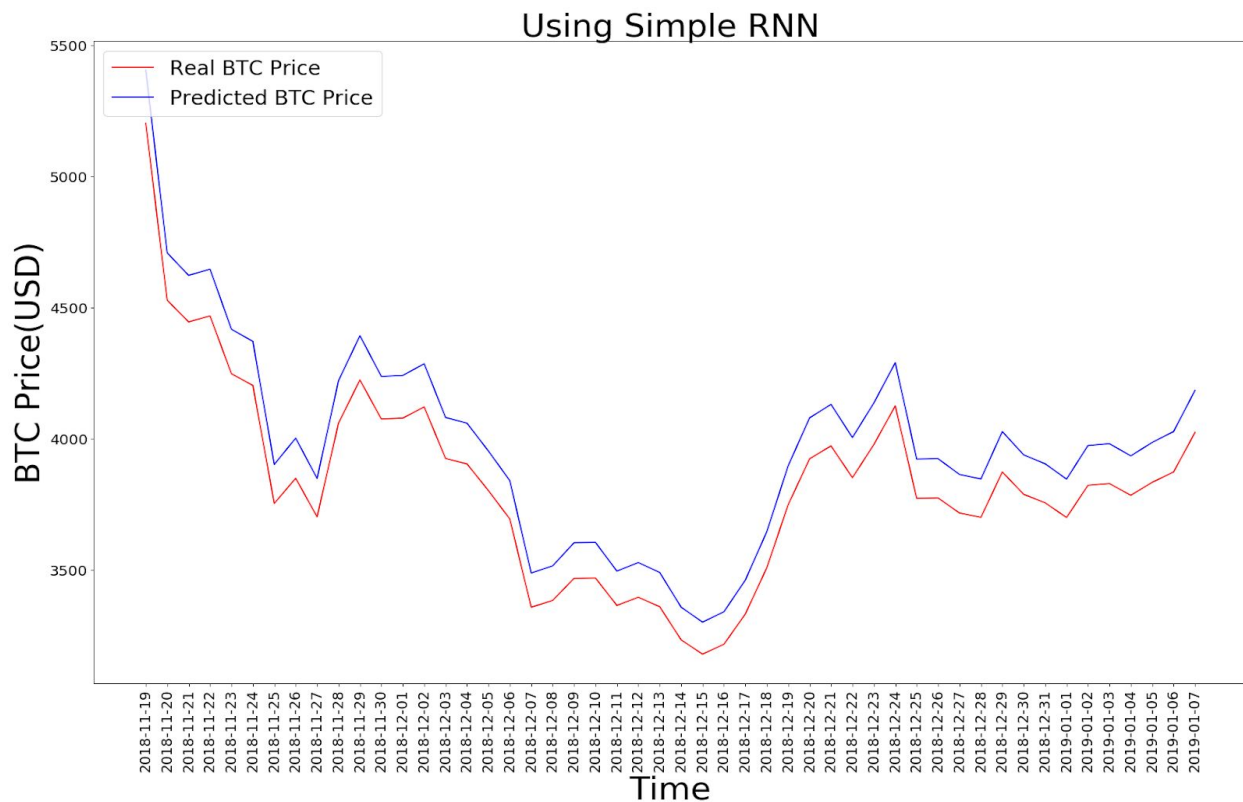


FIG: Plot of Bitcoin real and Predicted price by RNN

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.

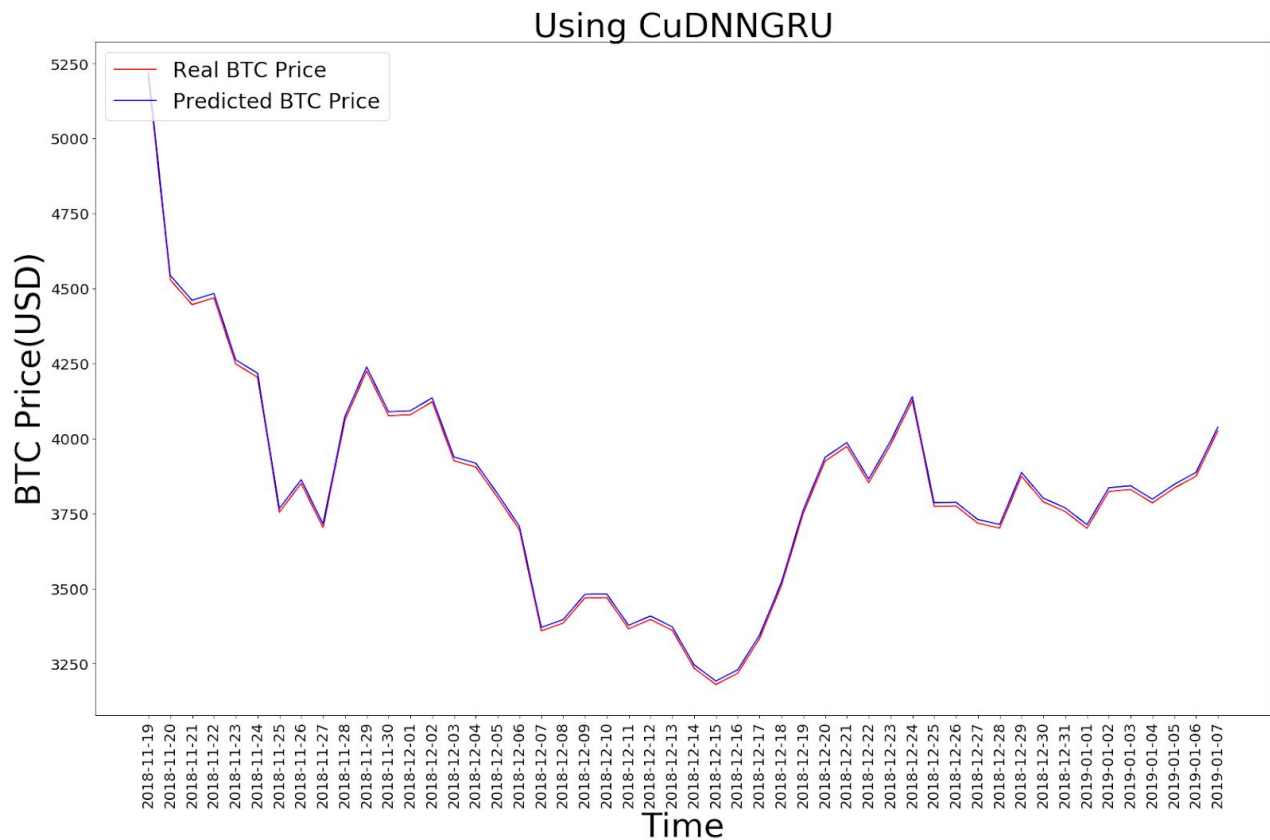


FIG: Plot of Bitcoin real and Predicted price by CuDNN GRU

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.