Building future cryptocurrencies price-movement predictor recurrent neural network

Deep Learning - data Science - UTEC

Diego Bugaiov
Michelle Moss
Pablo Dubourdieu
Carolina Gandolfo

August 2022

Content

1. Objective

The aim of this Project is to implement a recurrent neural network to predict 3 minutes into the future price movements of the cryptocurrencies based on the last 60 minutes data of that currency's prices along with another 3 cryptocurrencies prices and volumes.

In other words, the method is going to predict what is the price of the currency being traded at any given time.

2. Abstract

For this project, the four major cryptocurrencies were selected: Bitcoin (BTC), Litecoin (LTC), Ethereum (ETH) and Bitcoin cash (BCH). For the dataset, the price over time was tracked so that the result is the sequence over time of the price.

In this case, the price of any of these 4 currencies into the future is intended to be predicted, so the last 60 minutes of the price of each cryptocurrency is considered. The data collection is getting updated once every minute, so the collection is of 60 data.

The typical dataset is not in multiple sequences but a long sequence. To work with it, the data must be balanced, normalized -because for each coin the price and volume are different- and scaled.

The dataset may be found in the following link: dataset

3. Procedure

3.1 Data treatment

3.1.1 Get the data (Block #3 of the code)

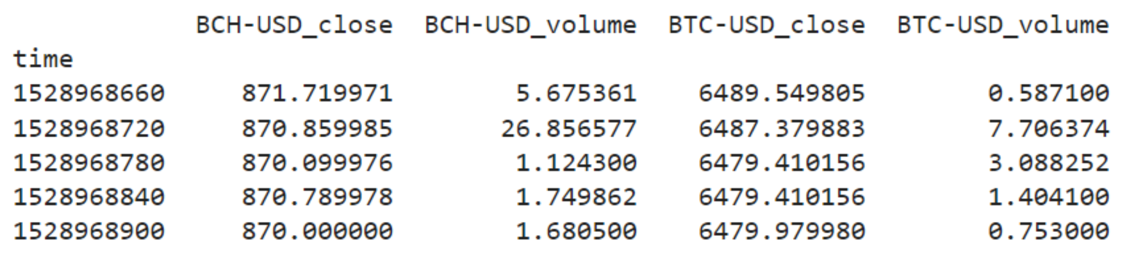
The dataset selected consists of 4 .csv files where information about low, high, open and close price, and volume are recorded every minute for over 1,500 hours for each of the four cryptocurrencies. Because price currency widely varies within a day, for the sake of simplicity, the "close price" column was chosen as an indicator of buying or selling, so this will be the key output variable to estimate. Now each data frame is separately prepared and the following step is to merge them all into one big dataframe. Since the data is recorded minute by minute for all of the files, the dataframe index is set to be "time" in minutes because it is common to all of the files.

The following step is to merge the four files into only one dataframe and decide which relevant variables are kept as inputs to the model.

3.1.2 Merge the data (Block #3 of the code)

Although there are five variables for each cryptocurrency as input variables to the model, it is decided to only consider "close price" and "volume" as the relevant inputs for each of the cryptocurrency (eight in total). As it was mentioned, the objective is to predict the "close price" for each of the cryptocurrencies from these eight variables, and therefore a model for each currency will be adjusted. This will help us to reduce the parameters for each model and also avoid unnecessary information about the markets as inputs.

In order to bring all the datasets together, each column must be identified with a unique name and for practicality they are named as "{cryptocurrency-USD}_close" and "{cryptocurrency-USD}_volume" where {cryptocurrency}=[BTC, LTC, ETH, BCH]. The columns are renamed and the dataframe head for the first rows is as follows:



```
            BCH-USD_close  BCH-USD_volume  BTC-USD_close  BTC-USD_volume
time
1528968660     871.719971        5.675361    6489.549805        0.587100
1528968720     870.859985       26.856577    6487.379883        7.706374
1528968780     870.099976        1.124300    6479.410156        3.088252
1528968840     870.789978        1.749862    6479.410156        1.404100
1528968900     870.000000        1.680500    6479.979980        0.753000
```

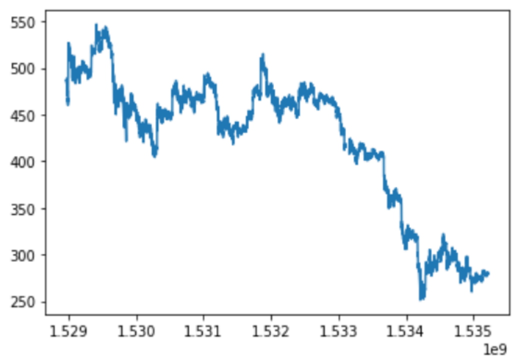
Now all the relevant file information has been prepared and merged in a single dataset.

3.1.3 Create targets (Block #5 of the code)

The dataframe is sequential data with timestamps, and because the future values for the different cryptocurrencies are aimed to be predicted, the targets or labels that are going to be used for each data point should. Any supervised machine learning problem 2 things are needed: sequences and targets.

Firstly, the target for each point is defined to be the "{cryptocurrency-USD}_close" three minutes forward in time. Accordingly, an extra column is constructed for each of the cryptocurrencies as a column for every minute. Moreover, this value for each data point is predicted based on the last 60 minutes backwards in time. So, for each of the points, a matrix of (60x8) must be constructed as input to the model. Secondly, a model is fitted for each cryptocurrency and therefore a variable is defined to indicate which cryptocurrency is being used. The following parameters are defined:

- ➔ Length of the sequence used for each prediction (SEQ_LEN): 60 minutes.
- ➔ Period which is going to be predicted (FUTURE_PERIOD_PREDICT): 3 minutes.
- ➔ Ratio chosen to make the prediction (RATIO_TO_PREDUCT): ETH-USD.



By now, the future price or "target" has been defined; for each moment, a prediction on the price in the next 3 minutes.

3.2 Defining training and validation datasets (Blocks #6 to #9 of the code)

3.2.1 Separate out out-of-sample data

In the case of time series data, it is not an option to just shuffle and then take a random 20 % as validation data and the other 80 % as training data. Moreover, the sequence order matters in this case and the data evolution is a crucial factor for future predictions and modeling. If the data is shuffled before it is passed through the model, we lose the evolution tendency and therefore it makes no sense to use a RNN.

3.2.2 Separate-out the validation data, out-of-sample data and training data

In the case of the data analyzed in this report, the last 5% of the historical data will be considered and separated as validation data. The reason is that the sequences are 60 minutes long, so there are 60 units in each sequence and the prediction is 3 minutes ahead (5 % of those 60 minutes backward). This option is the most realistic based on the prediction time that it is being considered. As a consequence, the threshold unit time is defined to be the last 5 % of the data and anywhere where the timestamp is greater than this value, it would correspond to validation data.

Challenge

While defining the percentage of the data to predict, there is a relationship between the range to predict and the accuracy of the predictions. In this currency series forecasting, there is no clear pattern or tendency so we had to sacrifice long ranged predictions for shorter and better ones.

3.3 Treating (Blocks #8 of the code)

Now that data is split, prices and volumes need to be converted into the same unit measurement to avoid biases. So, the relative change is being analyzed instead of absolute change to fit the neural network.

Then, sub-datasets for each of the points in the dataframe are built based on the last 60 minutes and the eight relevant variables selected above. In total, over 90,000 (60x8) matrices are created for training and validation.

3.4 Build and train the model (Blocks #10 and #11 of the code)

The model was built using the tensor Flow package from Python. A Sequential function is used and different layers are initially added based on different references. Then, layers modifications are adjusted and new layers are added based on trial and error.

The final RNN model for all the cryptocurrencies include a bidirectional layer to help with the parameter computations based on forward and backward paths, dropout layer to avoid node preferences by the network, CuDNNLSTM which mimics a LSTM layer but it is computed much faster than conventional LSTM, batch-normalization to normalize the output of the CuDNNLSTM layer and help with nonlinear function predictions, and dense layer to help making complex relation predictions. Moreover, a callback ModelCheckpoint was applied to only save the best model from training and the test loss was particularly considered.

Challenge

Understanding how to handle the dimensions of each layer. We had to understand input and output dimensions and adapt them to fit our model. The clause return_sequences=True

was both a solution and a source of issues. As a work around we used a Flatten layer until understanding how to correctly apply it.

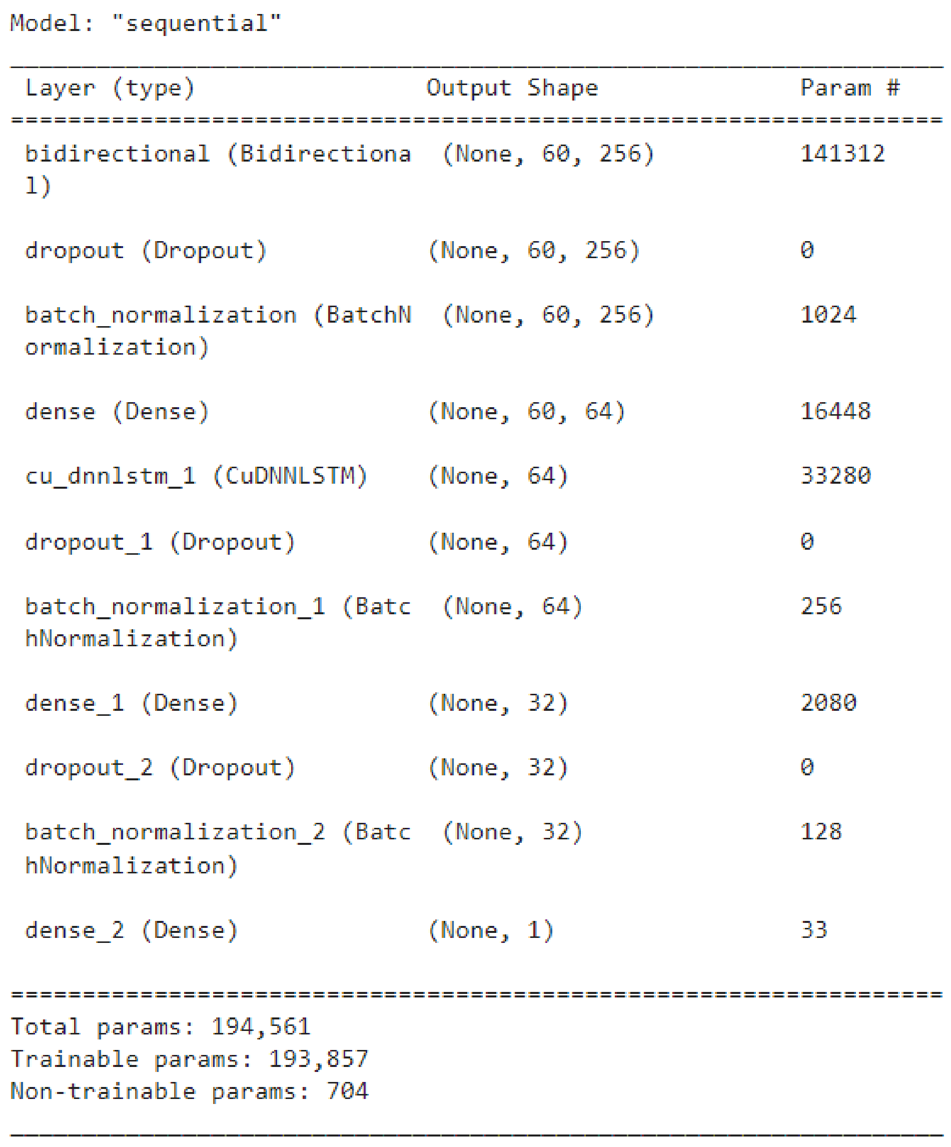
The activation layers are key to find different patterns in the series. We tried several activation functions, among others softmax, relu, linear, leakyRelu.

4. Discussion and conclusion (Blocks #12 to #18 of the code)

The results for Ethereum and Bitcoin currency are presented below.

Challenge

When deciding which model to apply we used several different sequences. In many cases the loss diverged or the model didn't learn at a good rate. We found that applying LSTM or CuDNNLSTM layers is key for the model to perform well in a series forecasting.



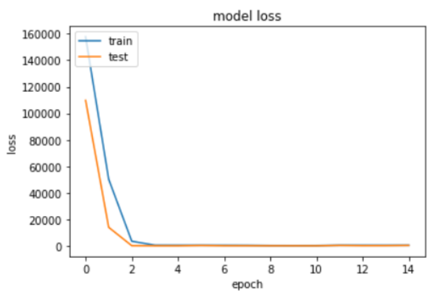
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 bidirectional (Bidirectiona  (None, 60, 256)          141312    
 l)                                                              
                                                                 
 dropout (Dropout)           (None, 60, 256)           0         
                                                                 
 batch_normalization (BatchN  (None, 60, 256)          1024      
 ormalization)                                                   
                                                                 
 dense (Dense)               (None, 60, 64)            16448     
                                                                 
 cu_dnnlstm_1 (CuDNNLSTM)    (None, 64)                33280     
                                                                 
 dropout_1 (Dropout)         (None, 64)                0         
                                                                 
 batch_normalization_1 (Batc  (None, 64)               256       
 hNormalization)                                                 
                                                                 
 dense_1 (Dense)             (None, 32)                2080      
                                                                 
 dropout_2 (Dropout)         (None, 32)                0         
                                                                 
 batch_normalization_2 (Batc  (None, 32)               128       
 hNormalization)                                                 
                                                                 
 dense_2 (Dense)             (None, 1)                 33        
                                                                 
=================================================================
Total params: 194,561
Trainable params: 193,857
Non-trainable params: 704
_________________________________________________________________
```

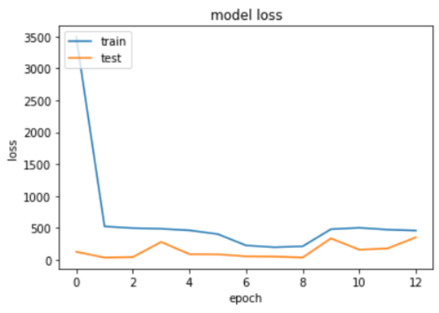
Model was fitted with the split data, using Adam optimizer and Mean Squared Error (MSE) as loss function. Then, using the model parameters, the training and validation labels are predicted and plotted to compare the results.

As a matter of simplicity, only these two currencies are compared to discuss the results. We can observe that the predictions for ETC are much more precise than BTC. The best MSE for ETC is roughly 420 whereas for BTC was 32,000. Moreover, the ETC model converges faster; it only needs two epochs to drastically decrease the loss value to its final value. However, BTC needs more than nine epochs to converge and the values are still worse than ETC. This could be because BTC is a more dynamic currency and its prediction is difficult to estimate. Although we can observe that almost 90 % of real data is between predicted values for BTC, the error is too large to be a precise estimation. On the other hand, ETC predictions are much more accurate, which implies less randomness in the market and therefore better estimations.

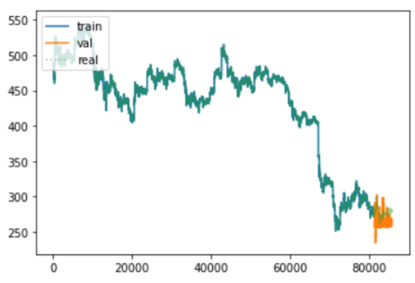
Another factor that could be affecting the results accuracy is the inputs selected to predict the close price. It is interesting to pass different combinations of inputs to the model, fit the parameters and compute the loss value for those combinations. It may be found that close price and volume are not the best input variables for predicting the close price and including other inputs would better estimate the real data.

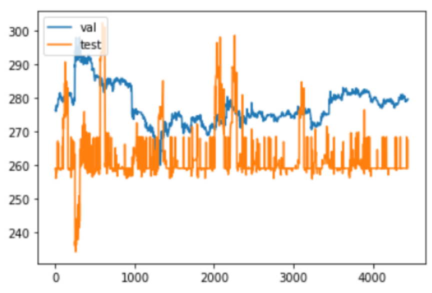
Loss function for training data and validation data for Ethereum currency



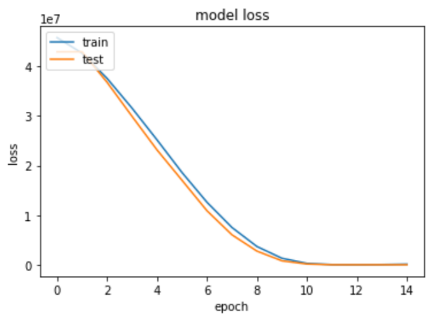


Training and validation label predictions compared to real data based on fitted model for Ethereum currency

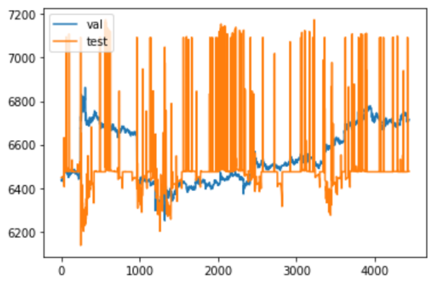


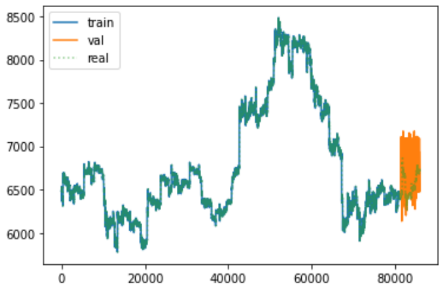


Loss function for training data and validation data for Bitcoin currency



Loss function for training data and validation data for Bitcoin currency





As conclusions it could be stated that:

- ➔ A Recurrent Neural Network was computed for each cryptocurrency and good "MSE" values were obtained.
- ➔ Close price values were predicted for each cryptocurrency with accurate results.

5. References

[1] Scientist, D. D. J. D., & Engineer, K. B. S. (2021, October 25). Machine learning mastery. Machine Learning Mastery. Retrieved August 8, 2022, from https://machinelearningmastery.com/

[2] *Home*. MLK - Machine Learning Knowledge. (2022, January 23). Retrieved August 8, 2022, from https://machinelearningknowledge.ai/

[3] Chablani, M. (2017, July 11). Batch normalization. Medium. Retrieved August 8, 2022, from https://towardsdatascience.com/batch-normalization-8a2e585775c9#:~:text=Using%20batch%20normalization%20allows%20us,difficult%20when%20creating%20deeper%20networks.

[4] Scikit-learn - Machine Learning in Python https://scikit-learn.org/ Accessed: August 2022.

[5] https://keras.io/ Accessed: August 2022