High Frequency Exchange Rate Forecasting using Deep Learning on Cryptocurrency Markets

Imanol Arrieta Ibarra
imanol@stanford.edu

Bernardo Ramos
bramos@stanford.edu

December 16, 2018

Abstract

In this work we develop a deep learning model for predicting high-frequency exchange rates of cryptocurrencies. Given their central role in digital assets, we focus on predicting 1-minute movements of Bitcoin-Ethereum. We find our model to have outstanding accuracy, and with it we briefly illustrate how our tool can be used in benefit of hedging and profitable trading strategies.

1 Motivation

Despite the widespread belief that the efficient market hypothesis holds, recent state-of-the-art methods have shown there are grounds to have reasonable performance in predicting financial returns. Following works as [4], [6], and [8], we wish to assess the extent of these models' performance in the non-traditional setting (i.e. non-equity, non-derivatives time series) of cryptocurrency exchange rates.

There are numerous reasons why predicting cryptocurrency exchange rates is a challenging problem. First, neither Deep Learning nor traditional Time Series Models in high-frequency equities data tend to have a performance that is superior to correctly predicting the sign above two-thirds of the time [9]. Second, cryptocurrency data is deemed largely volatile, making it harder to predict trends in very short time periods. Finally, there might not be a very strong predictor of movements in prices using market data.[1]

[1] In finance, indices such as the S&P 500 are good predictors of where individual stocks will move.

2 Related Work

With the advent of Deep Learning, traditional financial time series models are increasingly substituted in favor of models with superior performance, such as Recurrent Neural Networks. Because of this, the recent literature has been expanded with a number of efforts attempting to fit idiosyncrasies of financial data into deep learning paradigms.

Works following the above lines are illustrated by [2], who utilize Wavelet Transforms with stacked auto-encoders to generate features. We deviate, however, from such approaches since neither technique seems to improve predictions unless there are evident problems of high model bias. Moreover, it is noted by [12] that using wavelet de-noising can in fact harm model predictions, and they opt for using raw time series as inputs to the network. Were our models to show a noticeable bias problem we would consider producing higher-level features with such techniques as stacked auto-encoders.

3 Problem setup

Our aim is to predict relative movements in Bitcoin (BTC) and Ethereum (ETH) by forecasting the difference in their exchange rate ratio at the 1-minute frequency. Our application is intended to be used for trading purposes, therefore we use the Time-Weighted Average Price (TWAP) as the reference –or target– exchange rate.[2] The TWAP is computed as the average between the open, close, highest

[2] Interchangeably, we will refer to exchange rates as 'prices'.

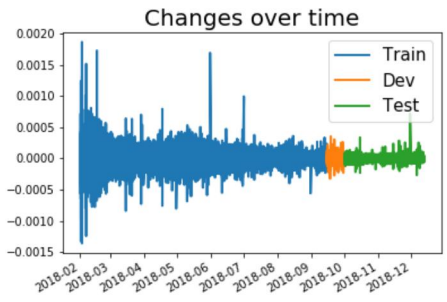

Figure 1: Returns over the study period.

and lowest price in a time window. It is a smoothed quantity that is crucial for robustly timing orders in accordance with price expectations.

3.1 Data

The data was obtained from a company called Binance, a cryptocurrency exchange which offers trade historic data (up to last year's September, when the company started) using their public API. Our data comprises the time series of exchange rates for Bitcoin (BTC), Ethereum (ETH), and Tether (USDT) at the 1-minute frequency from September 1, 2017 through October 31, 2018. An illustration of the data is provided in Figure 1, which shows the progression of returns over the study period.

3.2 Features and Data Imputation

For each minute window and each pair of cryptocurrencies, our data provides the open, close, high and low rates, as well as the volume of trades executed in that period. We provide the network with an easier representation of these series by computing first-order differences and adding both levels and differences into the inputs.

When no trades were executed or data was otherwise not available, we use a last-valid observation imputation scheme. To expand our data with potentially useful features, we append both an indicator of whether there were trades at any given point, and the number of timesteps since the last trade was executed. These becomes less of a problem as the liquidity in the crypto market increases, but we still correct for it because of the prevalence of these cases in the training data.[3]

Finally, a one-hot encoding for the hour of the day is also used, inspired by the fact that other financial markets experience heteroskedastic volatility throughout a trading day and day of the week.[4]

The data split was made so that validation data would comprise September 15, 2018 through September 30, and test data the entire month of October. Train, validation and test data respectively account for roughly 83%, 6% and 11% of the dataset, whose trainable volume totals little more than 60 variables and 380,000 observations.

4 Methods and Technical Approach

Following its success and popularity in time-series forecasting, our approach is to use an LSTM Recurrent Neural Network. We model this problem as a regression-type supervised learning to predict BTC-ETH one minute ahead, where inputs are histories of the time series of exchange rates as well as those features described in Section 3.2.

4.1 Architecture

The network we fit is a single LSTM module (of neuron size 32), followed by a 64-neuron fully-connected layer with Leaky ReLU activation and an output layer. Dropout (with a moderate rate of 0.3) is used after the first dense activation and before the non-linearity. Weights in the fully connected layers are regularized with an L^2-penalty (rate 0.0001). Both inputs of the model and outputs of the LSTM module are batch re-normalized, as it is shown that time-variant trends and volatilities make batchnorm less effective due to the strong dependence in mini-batch activations [7]. The sketch of the final architecture is shown in Figure 2. All hyperparameters –including the model architecture– were submitted to a random search process where 150 models were trained to yield a close-to-optimal

[3] Since we are working with time-series, we are constrained on the distribution for our train, dev and test sets. Because of the novelty of crypto markets, this causes the dev and test sets to have higher liquidity and lower volatility than the training set.

[4] Different from other markets, crypto has no time-constraints to be traded. This creates a tangible difference between the behaviour of prices when other markets are open, relative to when they are closed.

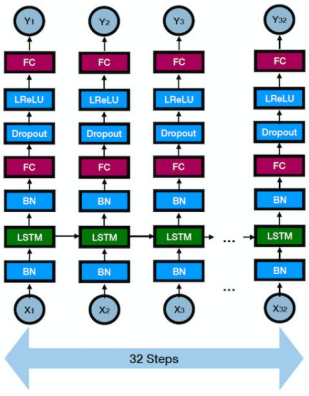

Figure 2: Network architecture.

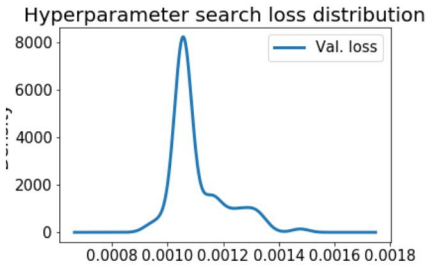

Figure 3: Hyperparameter search loss distribution.

configuration. Figure 3 shows the distribution of MSE for the hyperparameter configurations tried.

Training is performed in batches (of size 32) with sequence lengths of size 32, which requires clever arrangements of the input matrix to properly make use of Keras' stateful mode for LSTM modules. We found a sufficient number of burn-in steps –point at which the hidden state outputs meaningful history-embedded features– by empirically evaluating errors in the train set.

4.2 Computational Setup

We use TensorFlow [1] and Keras [3] as our deep learning library. The machines we used ran OS X without Tensorflow-compatible GPUs. Thus, all code was run on

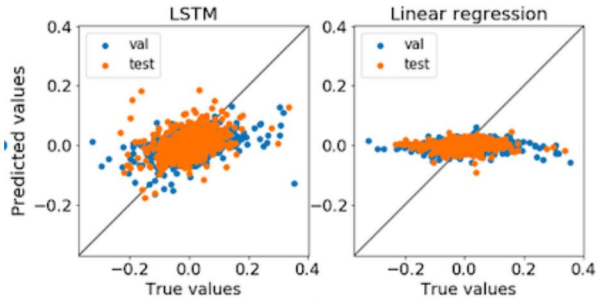

Figure 4: Scatterplot of validation set (true and predicted values) for LSTM and Linear Regression.

1.6 GHz Intel Core i5 CPUs. Model training takes around 15 seconds per epoch, totaling around 10 minutes for each end-to-end optimization run.

5 Results

Figure 4 shows the scatterplots of predicted versus the true values over the dev and test sets for LSTM and Linear Regression. All results shown are in a scale 1000 times larger than the true exchange rates, as this was found to yield model training easier by matching the output and target variances. Given the overall diagonal shape of the data cloud for LSTM (Left) along the 45-degree line, we can see the network fits out-of-sample data reasonably well. On the other hand Linear Regression (Right) shows a more reduced positive trend for the extremes of the distribution (where there is potential for higher rewards) .

To further analyze the performance of our model, we show in Table 1 three measures of goodness-of-fit. Notably, the hit rate –the proportion of times predictions have the correct sign– is around 60% for out-of-sample data. R^2 appears stable across the train, dev and test sets, which is remarkable given how different volatility behaves across these periods.[5]

[5]This difference between dev and test is not possible to correct for since we work with time-series analysis. However, we plan to expand this data in the future so that dev and test reflect similar trends.

	RMSE	R^2	Hit rate
train	0.07437	12.40%	59.44%
dev	0.03037	12.51%	59.23%
test	0.02319	12.19%	58.43%

Table 1: LSTM model performance measures.

5.1 Comparison to Baseline Models

Table 2 shows the performance of the LSTM model compared to the performance of two baseline models: linear regression and an autoregressive model with exogenous variables (ARX). All covariates and input features in these models correspond to the same input data fed into the LSTM network. [6]

When comparing to other models, results are mixed. On the one hand, R^2 sees an improvement from the linear model to the LSTM. In-sample ARX(1) predictions also seem to capture a good amount of the model variation given the high R^2, though this might change if the model had been trained on the train data to a point where it is comparable to the LSTM. [7] On the other hand, hit rate is rather low for the ARX model, whereas the linear and LSTM models show to be of the same order of magnitude.

Performance may not be uniformly superior to the benchmark due to the reduced number of variables included as features. With more input features (for example, including the space of all exchange rates), a gap between the linear model and the LSTM network would be more evident as the deep network would allow for complicated interactions in the cross-section of exchange rates. Such an analysis is left for future work.

[6] We could only test the ARX model insample which gives it an unfair advantage. However we find that even if this is the case it does no better in terms of profit to the LSTM

[7] Computational resources and the lack of optimized packages made it not possible to learn the ARX(1) model on the train data.

	RMSE	Hit rate	R^2
Linear	0.0234	**58%**	10.37%
LSTM	0.0232	**58%**	12.19%
ARX(1)	0.0224	46%	17.52%

Table 2: Model performance measures for the test set. *Note: ARX(1) measures were computed in-sample only for the test data. Its measures are therefore used as a reference only.

6 Trading Strategy

Previously we observed the LSTM model performs similarly to the linear model in determining the sign of the next movement. In trading, however, we are concerned about performance measures in specific parts of the prediction distribution. In this section, we show through a trading experiment how the Deep Learning model captures tail phenomena better than baseline models and is able to yield the highest profits.

Our simulation consists in having at each minute the opportunity to submit an order for either entering a long (buy) or short (sell) position on the BTC-ETH ratio. We define three trading strategies that used the predictions obtained from the previous models:

1. Threshold Strategy: For this strategy the agent buys (sells) Ethereum if at any given minute if the exchange rate is expected to go against (in favor) in a magnitude higher than a given threshold τ. The agent reverses the transaction by buying back Bitcoin when the model predicts the exchange rate will revert back.

2. Random Strategy: At each time period t the agent shorts or longs her position randomly.

3. Long-Short Strategy: Here, the agent alternates between buying and selling the base crypto. The rationale behind this strategy is that, in our data, roughly 60% of return movements are followed by a movement in the opposite direction.

Figure 5 shows the results for the simulation on the test portion of the dataset (i.e. October 1 through October 31, 2018). To make strategies' profit-and-loss (PnL) distributions comparable, all transaction magnitudes are performed with the equivalent of one unit of Bitcoin. The threshold chosen for all strategies was based on the median of changes experienced on the train data.

The only two strategies observed to be skewed to the right –and thus make profits– are the Long-Short strategy

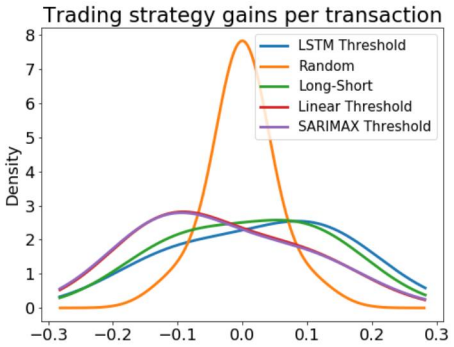

Figure 5: Profit and loss (PnL) distributions for each strategy.

Strategy	Mean	Std. dev.	% Positive
LSTM Thres.	**0.026**	0.108	**59.55**%
Linear Thres.	-0.035	0.102	36.39%
ARX Thres.	-0.036	0.104	36.03%
Random	-0.001	0.045	15.69%
Long-short	0.010	0.012	53.86%

Table 3: Simulation statistics for each strategy. Quantities are in BTC. '% Positive' refers to the proportion of transactions resulting in a strict profit.

and the LSTM Threshold Strategy. A closer look at Figure 5 reveals the LSTM strategy also has a lighter tail on the negative end. The linear and ARX threshold strategies have negative mean, which supports the idea that hit rate is not similar to the DL model for observations with large magnitude movements.[8]

Table 3 shows statistics on each transaction's profit and loss. We confirm the LSTM Threshold strategy yields the highest expected return, with a standard deviation comparable to the Linear and ARX-based strategies. Further, expected returns are negative or centered around zero for the random and baseline strategies. Finally, the highest-achieving rate of positive transactions –by a considerable margin– was the LSTM strategy.

[8]This phenomenon is visible from the comparison of the linear model to the LSTM scatterplots in Figure 4.

7 Conclusion

Our study showed that the advantages of Deep Learning over traditional methods are more intricate than just improvements in prediction performance. Although classical statistical tools (Linear Regression and ARX model) performed similarly in terms of prediction, Deep Learning forecasts outperformed these in terms of expected earnings following a threshold-based strategy. This, we find, is due to DL having superior predictive performance at the tails of the returns distributions. Classical methods, on the other hand, had better prediction accuracy close to the center of the distribution, where the earnings are lower.

Our results also showed that there are grounds for profitable trading strategies in the crypto markets. Although the returns we obtained are in the order of the "risk-free" interest rate (Treasury Bonds, for example), the fact that one can make a profit in expectation has interest of its own. In particular, due to the small amount of covariates we used in this analysis we think there is possibility for further improvements when working with richer datasets.

8 Future Work

With more time and computational power the following points could be further analyzed.

- Perform the analysis in periods with different volatility regimes. We were constrained by the availability of data in the Binance platform but are actively collecting more data to expand our analyses in the future.
- Crypto should further provide the chance to study trading behavior from the actors themselves. This would require a lot of work in combining the trading datasets with the ledger (the crypto datasets) information.
- Expand our analysis with more data to further highlight the capacity of deep learning frameworks to learn complex feature interactions and boost performance. Possible expansions include high frequency stock market data to estimate the relationship between these two markets, as well as exchange rates from other cryptocurrencies.

References

[1] ABADI, M., AGARWAL, A., BARHAM, P., BREVDO, E., CHEN, Z., CITRO, C., CORRADO, G. S., DAVIS, A., DEAN, J., DEVIN, M., GHEMAWAT, S., GOODFELLOW, I., HARP, A., IRVING, G., ISARD, M., JIA, Y., JOZEFOWICZ, R., KAISER, L., KUDLUR, M., LEVENBERG, J., MANÉ, D., MONGA, R., MOORE, S., MURRAY, D., OLAH, C., SCHUSTER, M., SHLENS, J., STEINER, B., SUTSKEVER, I., TALWAR, K., TUCKER, P., VANHOUCKE, V., VASUDEVAN, V., VIÉGAS, F., VINYALS, O., WARDEN, P., WATTENBERG, M., WICKE, M., YU, Y., AND ZHENG, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 3

[2] BAO, W., YUE, J., AND RAO, Y. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PloS one 12*, 7 (2017), e0180944. 1

[3] CHOLLET, F., ET AL. Keras, 2015. 3

[4] GULLAPALLI, S. Learning to predict cryptocurrency price using artificial neural network models of time series. 1

[5] HOCHREITER, S., AND SCHMIDHUBER, J. Long short-term memory. *Neural computation 9*, 8 (1997), 1735–1780.

[6] HUISU, J., LEE, J., KO, H., AND LEE, W. Predicting bitcoin prices by using rolling window lstm model. 1

[7] IOFFE, S. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. In *Advances in Neural Information Processing Systems* (2017), pp. 1945–1953. 2

[8] KINDERIS, M., BEZBRADICA, M., AND CRANE, M. Bitcoin currency fluctuation. 1

[9] MCNALLY, S., ROCHE, J., AND CATON, S. Predicting the price of bitcoin using machine learning. In *Parallel, Distributed and Network-based Processing (PDP), 2018 26th Euromicro International Conference on* (2018), IEEE, pp. 339–343. 1

[10] SERMANET, P., EIGEN, D., ZHANG, X., MATHIEU, M., FERGUS, R., AND LECUN, Y. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229* (2013).

[11] SILVER, D., HUANG, A., MADDISON, C. J., GUEZ, A., SIFRE, L., VAN DEN DRIESSCHE, G., SCHRITTWIESER, J., ANTONOGLOU, I., PANNEERSHELVAM, V., LANCTOT, M., ET AL. Mastering the game of go with deep neural networks and tree search. *Nature 529*, 7587 (2016), 484–489.

[12] WANG, L., AND GUPTA, S. Neural networks and wavelet de-noising for stock trading and prediction. In *Time Series Analysis, Modeling and Applications*. Springer, 2013, pp. 229–247. 1

Code

The private GitHub repo can be found at https://github.com/bernardoramos/cs230project.