A Deep Learning Architecture for Intelligently Trading Cryptocurrencies

Gautam B. Singh
Professor
Oakland University
Rochester, Michigan, USA.

Abstract

This paper proposes an architecture that utilizes a deep learning for building a cryptocurrency trading platform driven by software robots. The challenge in cryptocurrency trading is the inherent volatility in their price making it harder to predict future trends. The proposed architecture aims to address this problem by continually updating the learning parameters used by the software bots in making their predictions enabling the bots to effectively utilize recent price fluctuations and historic patterns in making a buy, sell or hold recommendations. The bots generate a trading statistic that lends itself for use by a spectrum of portfolio management strategies ranging from those looking for aggressive growth to those that seek wealth preservation.

Keywords

Deep neural networks, Cryptocurrencies, Portfolio management, Market volatility, Algorithmic trading, and Auto encoders.

1. Background

Deep learning is a sub-field of artificial intelligence that aims at creating large neural networks that can make accurate data-driven decisions. This is particularly appealing in situations where we may have access to large quantities of data but lack an understanding of relationships between data components either because of data being complex, or because of its sheer volume, or both [7].

The generalized architecture of a deep-learning system is inspired by the organization of the brain. In a manner similar to our brain learning from the experiences and associated rewards, the deep learnings system associate a reward when the system produces a correct response corresponding to a training input data. As a larger number of training data sets are fed into the deep learning system, it learns a set of model parameters that maximize its reward [7].

Of late cryptocurrencies have gained considerable prominence and have begun to be a part of many robust investment portfolios. The last decade has witnessed a boom in the cryptocurrency market. Cryptocurrencies are decentralized and differ from fiat currencies in many important ways. Most significantly, cryptocurrencies utilize a new distributed payment system based on block-chains. And cryptocurrencies have a basis in cryptography that ensures anonymity, security, and low cost of transactions.

Currently in the cryptocurrency market, the most notable currency is Bitcoin, which was created by Satoshi Nakamoto in 2009. By 2018, the total market capitalization of Bitcoin surpassed 116 Billion US Dollars (USD) according to coinmarketcap.com.

Over a thousands of other new cryptocurrencies have emerged since then most notable being including Ethereum (ETH) and Ripple (XRP).

Cryptocurrencies are subject to extensive volatility and fluctuations. Many factors, including market uncertainty, investors' expectations, and emotion are generally responsible for such extreme cryptocurrency fluctuations. Analysis of the performance of cryptocurrencies such as Bitcoin fails to establish any relationships, especially volatility connectedness or spill overs, between the prices of cryptocurrencies. [12].

The rapid generation and volume of cryptocurrency transactions make it very difficult to apply any traditional forecasting models to cryptocurrencies. In order to overcome this problem, this paper proposes a framework that utilizes deep learning for predicting trends in cryptocurrency prices. Deep learning systems offer an advantage that the exact relationship between data elements need not be well understood a-priori. This in turn is considerably appealing from the standpoint of predicting concurrency volatility particularly when the casualty of these changes is not understood.

An architectural framework is proposed for building an autonomous predictive framework that continually learns model parameters to track this price volatility.

2. Data Management Framework

At the core of the system is a database which is populated by a continuous scraping of cryptocurrency transactional data. Presented below in Fig. 1 is a schema of this database.

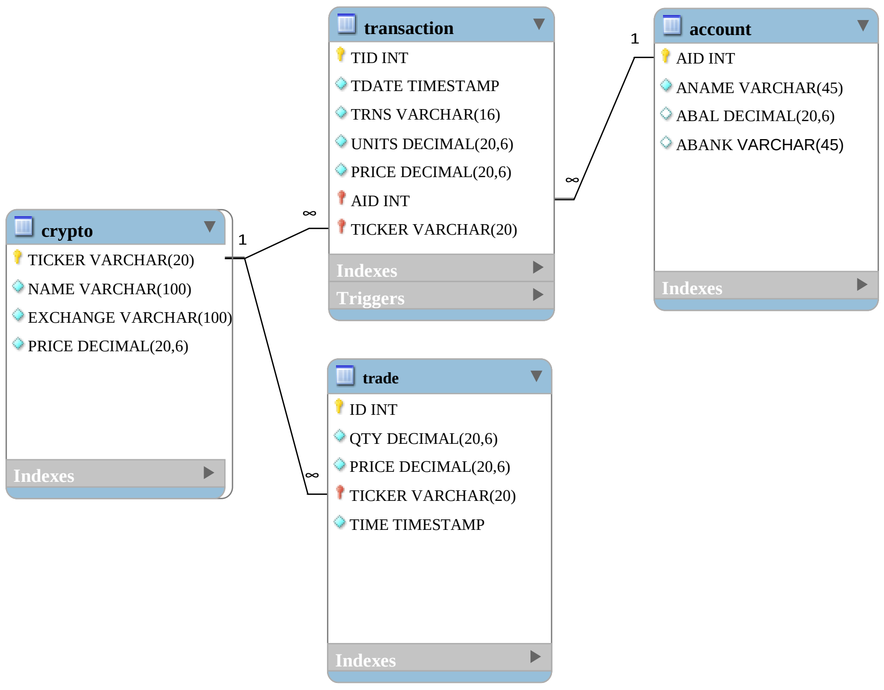

Figure 1: Database Schema

As illustrated in Fig. 1, the specifics of the cryptocurrency being are being tracked in the table crypto which is indexed on the specific exchange as well as the symbol of the cryptocurrency. Generally, however, regardless of the exchange being used for the purposes of tracking in the database, the actual cryptocurrency trades are recorded in a blockchain and therefore are not dependent on the specific exchange being used to capture the data.

The table trade is where the data of the specific cryptocurrencies of interest is being tracked. Specifically, every time there is a trade placed for a cryptocurrency and recorded on the blockchain, a record is created in this table. This table is primarily used for driving deep learning models. Details of the trade, including its timestamp, cryptocurrency ticker, the quantity exchanged, and the price at which the exchange occurred is recorded.

The other two tables, namely transaction, and account are not directly related to building the deep learning model being developed for unraveling and tracking the volatility in cryptocurrency prices. Rather, these are designed to effectively track and evaluate the performance of robot - driven portfolio management strategies. Since their role in the current discussion is rather limited, these will not be discussed further.

2.1 Automated Downloads of Historical Trading Data

The frequency of downloading trade data will depend upon the restrictions imposed by the exchange being utilized. Our system is constrained by the Kraken exchange which limits the number of records that may be downloaded at a time to be 720. Consequently, a frequency of downloading every 8-hours works well for our robots. The following Python code performs such a download using a script that wakes up every 8-hours, bulk downloads the transactions at a resolution of 1 minute, and imports these transactions into the database.

```
interval_list = [1, 5, 15, 30, 60, 240, 1440, 10080, 21600]
max_lookback_entries = 720

for interval in interval_list:
    first_ts, last_ts = get_first_last_timestamp (db, 'BITCUSD')

    lookback_in_sec = int (max_lookback_entries * interval * 60)
    since = xchng.get_timestamp () - lookback_in_sec
    if since > first_ts:
        since = last_ts

    df, lst = get_ohlc (xchng, 'BITC', 'USD', interval, since)
    write_dataframe_to_db (db, df, 'BITCUSD')
```

The automated script listed runs as a cron-job every 8-hour hours and downloads the latest trading data into our database shown in Figure 1. Most exchanges limit the quantity of information they allow software bots to download at a time. The script will therefore need to modified to match the specific restrictions of an exchanged being used.

Given that there is a limit on the number of records that the exchange allows to be downloaded, the automated download script in this case downloads these records at various resolutions. At the finest level of granularity, trading information is available at an interval of 1 minutes. And, at the coarsest level of granularity, trading information is available at 21,600 minutes. The following table lists the extent of trading data look-back possible if the data is downloaded directly from the exchange.

Table 1: Kraken Crypto Exchange: Look-back Time Intervals for Various Granularity Levels

Interval Granularity (minutes)	Block Size (fixed)	Duration
1	720	12 hours
5	720	60 hours (2.5 days)
15	720	180 hours (7.5 days)
30	720	15 days
60	720	30 days
240	720	120 days
1,440	720	720 days (1.97 years)
10,080	720	5,040 days (13.8 years)
21,600	720	10,080 days (29.6 years)

Cryptocurrencies being a rather new phenomenon, the data points at two coarsest levels of granularity – 10,080 and 21,600 minutes are not very relevant for our models since that historical data did not exist at this time. As a practical matter, since we don't want to miss any data points, the automated download robots initiate trade data downloads every 8 hours and remove any duplicated records from the database. Also, as a practical matter, all trades are indexed using the UNIX timestamp which is effectively what is utilized by the blockchain to make the process uniform across the globe.

3. Deep Neural Network Architecture

Deep learning is a study of artificial neural networks (NNs). A conventional neural network (NN) requires real-valued activation where we can change the weights to make NNs perform the desired task [3]. However, training a neural network is the lengthy process. Back-propagation is a prominent gradient descent strategy in neural networks since 1980. However, because back-propagation employs local gradient information with a random start point, it can get stuck in local optima. Also, if the training data is too small, NNs may overfit.

With an algorithm for layer-wise greedy learning, Hinton launched the era of deep learning [4]. Hinton recommended unsupervised learning before layer-by-layer training. Before exporting features to the next tier, features from the inputs are deleted. Using features collected from the inputs, the data dimension is thus reduced. In the following layer, all samples are labeled and the network is fine-tuned using the labeled data. Second, the pre-training technique preceding unsupervised learning assigns the network non-random initial values. As a result, the convergent rate is increased.

In 2016, Google's DeepMind team staged a Go game in South Korea between AlphaGo and Lee Se-dol, one of the world's greatest players. AlphaGo defeated Lee Sedol 4:1 using deep learning techniques [1]. Deep learning algorithms have also shown great performance in predicting the activity of potential medicinal drugs and the effect of non-coding DNA mutations on gene expression.

Neural networks with deep architectures have built a robust foundation for supervised learning. A deep learning method is built of many layers, each representing a non-linear information processing unit. Deep Neural Networks or DNNs can represent more complex functions by increasing the number of layers and units in a single layer. Deep learning can help people construct mapping functions with enough labeled training data and correct models [9].

There are four main deep learning architectures: Restricted Boltzmann Machines (RBMs) [4], Deep Belief Networks (DBNs) [2], Autoencoders (AE) [8], [11], and Convolutional Neural Networks (CNNs) [6]. RBMs are generative stochastic artificial neural networks that can learn a probability distribution from their inputs. RBM can be used in deep learning. Stacking RBMs and fine-tuning the resulting deep network using gradient descent and back-propagation creates deep belief networks proposed by Hinton in 2006. DBN may be used to tackle unsupervised learning problems to reduce feature dimensionality and supervised learning problems to develop classification or regression models. A DBN is trained in two stages: layer-by-layer and fine-tuning. After unsupervised training, the parameters of DBN are fine-tuned using error back-propagation methods.

An autoencoder (AE) is an unsupervised learning technique that codes a dataset to reduce dimensionality. The AEs have been at the forefront of ANN research for decades. Bourlard and Kamp discovered in 1988 that a multi-layer perceptron (MLP) in auto-association mode might compress data and reduce dimensionality in information processing [5].

The encoder function first converts the input data to an abstract representation, which is subsequently transformed back to the original format. It is taught to encode input into a representation that can be recovered from its representation. In this procedure, the AE tries to approach the identity function. The AE may filter out irrelevant information and extract useful features continually throughout propagation. Also, because the input vector is reduced in size during coding, the learning process is more efficient.

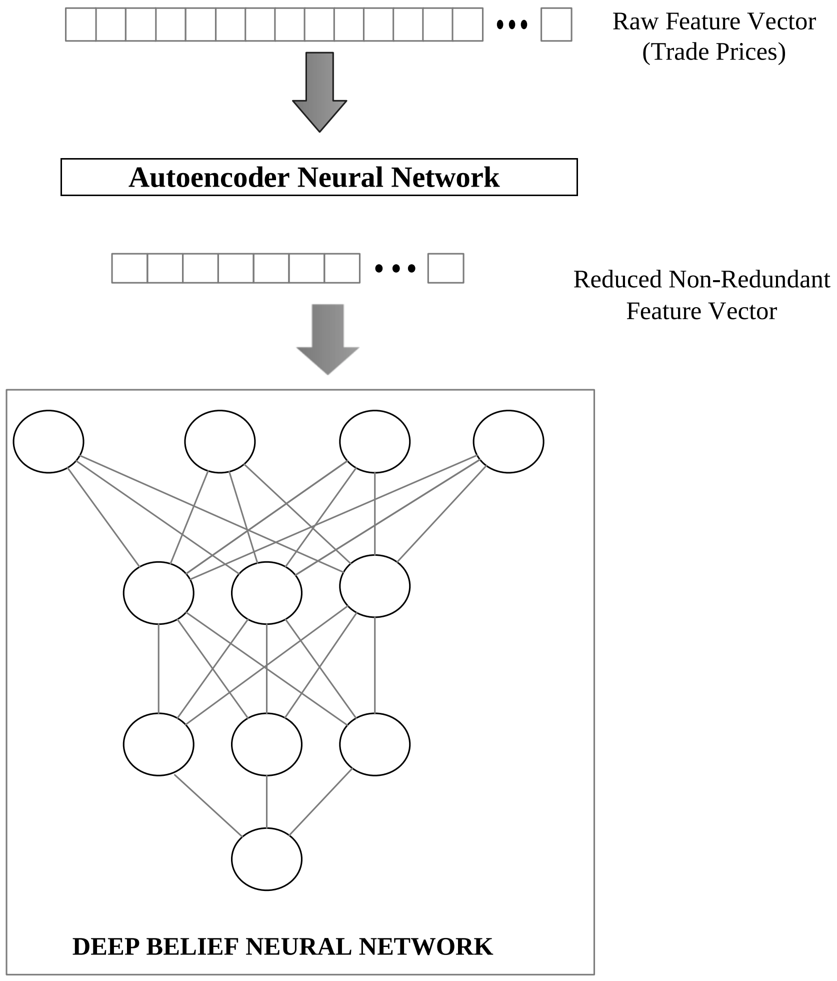

Figure 2: Deep Neural Network Architecture

4. Robot Learning and Trading

As illustrated in Figure 2, our cryptocurrency framework is designed to combine the use of Autoencoders and Deep Belief Networks to learn price predictions. This is different from the proposed stochastic models which process cryptocurrency data as a neural network with Long Short Term Memory (LSTM) [10]. The approach utilized in our framework does an unsupervised gathering of long and short-term price variations and combines them using an autoencoder and then trains a Deep Belief Network. A large volumes of data historical trade data in multiple resolutions are combined into a linear N-dimensional vector: $\{P_K^R\}$ where there are K price data points for resolution R, and ΣRΣK = N. For example, we can create a 120-dimensional vector by combining data elements from historical price trading information.

Table 2: Composition of Price Vectors

Resolution (R)	**Price Data Samples (K^R)**	**Look Back Period**
1 minute	60 samples	1 hours
5 minute	24 samples	2 hours
15 minute	12 samples	3 hours
30 minutes	8 samples	4 hours
60 minutes	6 samples	6 hours
240 minutes	6 samples	24 hours
1440 minutes	4 samples	96 hours
	120 samples	136 hours = 5 days and 16 hours

As may be observed in this formulation, more samples are taken from the recent trading trends where the DNN will be trained to capture more the fluctuations in the trading price within a backdrop of the longer-term pricing trends. The idea will be to let the neural network learn the weights of how the current price is influenced by (i) the short-term price fluctuations, and (b) the longer-term changes in price.

4.1 Target or Predicted Variable

The training process for a DNN takes in an input vector such as the one described above and tries to predict a target variable. This can be tricky since there is considerable volatility in cryptocurrency prices. Our system adopts an idea of look-ahead for a Price Trend Duration (T_D) where the price of the cryptocurrency is looked ahead for this predefined period of time to determine its average value, P_{LA}. This is compared with a look back for the same period (T_D) to compute its corresponding average value, P_{LB}. The differential in the two average values is used to determine the overall trend in pricing. Thus, if the Price Trend Duration of T_D has been chosen, and the D_{NN} is being trained at time t, the two prices P_{LB} and P_{LA} are calculated as follows:

$$P_{LB} = \frac{\sum_{i=t}^{t-T_D} P_i}{T_D} \tag{4.1}$$

$$P_{LA} = \frac{\sum_{i=t}^{t+T_D} P_i}{T_D} \tag{4.2}$$

The predicted or target variable is used as signum function which aims to evaluate the overall change being upward or downwards in the price. That is, an average trend will be deemed as *upward* if the average difference between the P_{LA} and P_{LB} is above a predefined threshold. This could then the tantamount to a decision to **Buy.** Conversely, a negative value for the difference between the P_{LA} and P_{LB} below a threshold value indicates a *downward* tend and tantamount to a Sell decision. Otherwise, the decision will be to **Hold.** Equation (4.3) below mathematically provide a formulation for a trend statistic **O** that is used for comparing to a portfolio goal driven threshold τ_α.

$$O = \frac{P_{LA} - P_{LB}}{P_{LA} + P_{LB}} \geq \tau_\alpha \Rightarrow \textbf{Sell} \tag{4.3}$$

$$O = \frac{P_{LA} - P_{LB}}{P_{LA} + P_{LB}} \leq -\tau_\alpha \Rightarrow \textbf{Buy}$$

While Equation (4.3) offers a strategy for robot-based management of portfolios, the exact actions taken for a specific portfolio will be further determined by the portfolio management strategy adopted for that portfolio. Such as strategy is adopted by setting an appropriate value for τ_α. For example, a specific portfolio rules will translate the value of τ_α into how much and whether to buy or to sell at all. These decisions can now be tied to a uniform variable and managed as another set of rules based on the aggressiveness and wealth preservation objectives of a given portfolio.

5. Tuning the Learning Algorithm

We utilize the parameter **O** as the target variable for prediction by the DNN. The process of training utilizes the sample generation as shown in Figure 3. Samples are generated by selecting a time points and selecting a set of trading prices as configured by the price vector illustrated in Table 2. Next, the value of **O** is computed as specified by Eq. 4.3. The price vector and the corresponding **O** values are computed for large number of overlapping samples where neighboring samples overlap by a predetermined time unit, typically 1 hour. In this manner, each sample comprises 60 new price values. The new price values are the high granularity values corresponding to 1 hour of trading data captured at 1-minute interval.

As the system goes online, it can continually learn to adapt to changes encountered in trading prices. Such a relearning process is initiated at predetermined intervals, typically every 24 hours. This ensures that emerging trends that have not been seen by the D_{NN} before are used to adjust the model weights and which will continue to become a better predictor of the statistic **O** continually as new trading data becomes available.

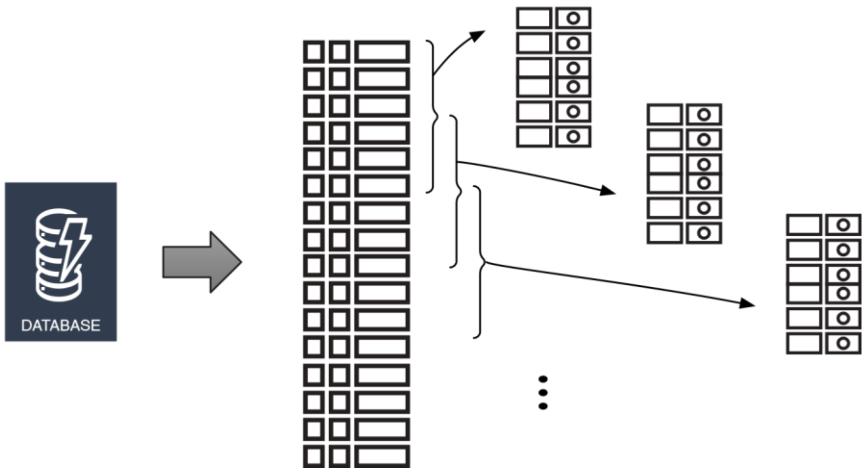

Figure 3: Generating Training and Test Samples from Database

6. Conclusions

The paper provided a framework for developing a bot-driven autonomous platform for trading cryptocurrencies. The system is designed to be self-learning and continually improves its performance by adjusting the parameters of DNN utilized so that the system can accurately predict the value of the statistic **O**. The system may be adapted to generate other statistics, in addition to the statistic **O**, that are used by a downstream portfolio management systems which make a decisions tied to specific objectives for managing portfolio conservatively, or aggressively, as desired by the end user.

7. References

1. D. Silver, A. Huang, C. J. Maddison, *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
2. D. Yu and L. Deng, "Deep learning and its applications to signal and information processing [exploratory dsp]," *IEEE Signal Processing Magazine*, vol. 28, no. 1, pp. 145–154, 2010.
3. G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

4. G. E. Hinton, S. Osindero, and Y.-W. Teh, “A fast learning algorithm for deep belief nets,” *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.
5. H. Bourlard and Y. Kamp, “Auto-association by multilayer perceptrons and singular value decomposition,” *Biological cybernetics*, vol. 59, no. 4, pp. 291– 294, 1988.
6. I. Arel, D. C. Rose, and T. P. Karnowski, “Research frontier: Deep machine learning–a new frontier in artificial intelligence research,” *IEEE computational intelligence magazine*, vol. 5, no. 4, pp. 13–18, 2010.
7. J. D. Kellenher, *Deep Learning*. MIT Press, 2019.
8. J. Schmidhuber, “Deep learning in neural networks: An overview,” *Neural networks*, vol. 61, pp. 85–117, 2015.
9. L. Deng, “Three Classes of Deep Learning Architectures And Their Applications: A Tutorial Survey,” *Apsipa Transactions on Signal and Information Processing*, Vol. 57, P. 58, 2012.
10. P. Jay, V. Kalariya, P. Parmar, S. Tanwar, N. Kumar, and M. Alazab, “Stochastic Neural Networks for Cryptocurrency Price Prediction,” *Ieee Access*, Vol. 8, Pp. 82804–82818, 2020.
11. P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, “Extracting and Composing Robust Features with Denoising Autoencoders,” In *Proceedings of the 25th International Conference on Machine Learning*, 2008, Pp. 1096–1103.
12. S. Corbet, B. Lucey, A. Urquhart, and L. Yarovaya, “Cryptocurrencies as a Financial Asset: A Systematic Analysis,” *International Review of Financial Analysis*, vol. 62, pp. 182–199, 2019.