Alonso-Monsalve, S., Suárez-Cetrulo, A.L., Cervantes, A., Quintana, D. (2020). Convolution on neural networks for high-frequency trend prediction of cryptocurrency exchange rates using technical indicators. *Expert Systems with Applications*, 149,113250.

DOI: 10.1039/c3lc51419f

Convolution on Neural Networks for High-Frequency Trend Prediction of Cryptocurrency Exchange Rates Using Technical Indicators

Saúl Alonso-Monsalve[a,c], Andrés L. Suárez-Cetrulo[b,c], Alejandro Cervantes[c,*], David Quintana[c]

[a] *CERN, Geneva, Switzerland*
[b] *Centre for Applied Data Analytics Research, University College Dublin, D04 V2N9 Dublin, Ireland*
[c] *Department of Computer Science and Engineering, Universidad Carlos III de Madrid, Avda. Universidad 30, 28911 Leganes, Spain.*

Abstract

This study explores the suitability of neural networks with a convolutional component as an alternative to traditional multilayer perceptrons in the domain of trend classification of cryptocurrency exchange rates using technical analysis in high frequencies. The experimental work compares the performance of four different network architectures -convolutional neural network, hybrid CNN-LSTM network, multilayer perceptron and radial basis function neural network- to predict whether six popular cryptocurrencies -Bitcoin, Dash, Ether, Litecoin, Monero and Ripple- will increase their value vs. USD in the next minute. The results, based on 18 technical indicators derived from the exchange rates at a one-minute resolution over one year, suggest that all series were predictable to a certain extent using the technical indicators. Convolutional LSTM neural networks outperformed all the rest significantly, while CNN neural networks were also able to provide good results specially in the Bitcoin, Ether and Litecoin cryptocurrencies.

Keywords: Cryptocurrencies, Neural Network, Finance, Technical Analysis, Deep Learning

*Corresponding author
Email addresses: saul.alonso.monsalve@cern.ch (Saúl Alonso-Monsalve), andres.suarez-cetrulo@ucd.ie (Andrés L. Suárez-Cetrulo), acervant@inf.uc3m.es (Alejandro Cervantes), dquintan@inf.uc3m.es (David Quintana)

1. Introduction

Cryptocurrencies are a kind of digital assets based on cryptographic protocols and technologies, such as the blockchain, that run on decentralized networks and make transactions secure and difficult to fake. These, which are emerging as an alternative to traditional centralized currencies, have attracted significant attention in recent years due to the blockchain ecosystem and the high volatility of their exchange rates (Li & Wang, 2017; Vidal-Tomás & Ibañez, 2018; Nakano et al., 2018).

Financial prediction is a domain full of challenges. Market data is often characterized by the presence of noise, a high degree of uncertainty, and hidden relationships (Huang et al., 2005). Apart from the use of raw prices and volumes using traditional statistical methods, the prediction of price movements can be approached using fundamental and technical indicators (Lo et al., 2000; Tay & Cao, 2001). In this domain, Machine Learning algorithms have grabbed the interest of academics and practitioners due to their ability to capture nonlinear relationships in the input data and predict price movements without relying on traditional assumptions on their statistical properties nor introducing human bias (Atsalakis & Valavanis, 2009; Hsu et al., 2016). These have benefited the automation of algorithmic trades in financial instruments at very high speeds. In this context, High-Frequency Trading (HFT) helps traders hold positions for short periods of time and earn their profits by accumulating tiny gains on a large number of transactions (Huang et al., 2019). On the academic side, there is a wide literature supporting the relevance forecasting at high frequencies (Huang et al., 2019; Dempster & Leemans, 2006; Daníelsson & Love, 2006; Zafeiriou & Kalles, 2013; Nelson et al., 2017; Borovkova & Tsiamas, 2018; Nakano et al., 2018; Chong et al., 2017; Shintate & Pichl, 2019).

The use of HFT is especially appealing in the context of crypto-currency exchange rates, due to their intraday volatility. As a consequence, many HFT firms have started operating and offering collocation of services in cryptocurrencies for institutional investors. The development of cryptocurrencies has coincided with a renewed interest in Neural Networks. This technique, despite being decades old, has gained considerable notoriety due to the current level of maturity reached by some feature learning frameworks, the success of deep learning the last few years in image recognition, and the high scalabil-

ity of their algorithms by using GPUs. This interest has been spread as well to the financial domain, with an increase of research on stock market price prediction using different deep neural network architectures for short-term and intraday technical trading (Chong et al., 2017; Lahmiri & Bekiros, 2019; Mallqui & Fernandes, 2019).

In this context, the aim of this paper is contributing with new evidence on the suitability of using neural networks with a convolutional component to make intraday trend classification for cryptocurrencies based on technical indicators. More specifically, we will benchmark Convolutional Neural Networks (CNN), hybrid CNN-LSTM networks (CLSTM), Multilayer Perceptrons (MLP) and Radial Basis Function Neural Networks (RBFNN) on intraday data for Bitcoin, Dash, Ether, Litecoin, Monero and Ripple.

Given its nature, the study is more focused on the performance of the instrumental in the domain, and less on the design and implementation of profitable trading systems. The latter practical application would require, in addition to the implementation of a decision component, controlling for aspects like time management, liquidity issues, or transaction costs, among others.

The rest of the document is structured as follows: first, we will provide an introduction to the relevant literature. That will be followed by a brief description of the network architectures compared in the study. The next section will be devoted to describing the experimental design. After that, we will cover the experimental results and, finally, the last one will be reserved for a summary and conclusions.

2. Literature Review

2.1. Cryptocurrencies

Cryptocurrencies have attracted the attention of investors and regulators since they were first proposed (Nakamoto, 2009). Their popularity relies on their peer-to-peer system, their ungoverned nature, and their low transaction costs. This has led to a surge in trading volume, volatility and price on exchanges. Either as a cause or an effect of this, they have become mainstream in media. Glaser et al. (Glaser et al., 2014) and Baek and Elbeck (Baek & Elbeck, 2015) defend that cryptocurrencies constitute a new asset class with more elements in common to speculative commodities than currencies, as their value is not based in any tangible asset. This has also raised the interest of the academic community.

There is a wide variety of recent studies covering financial aspects of different cryptocurrencies such as market efficiency (Vidal-Tomás & Ibañez, 2018); price volatility and study dynamic relationships between them (Katsiampa, 2017; Corbet et al., 2018); transaction cost (Kim, 2017); price clustering (Urquhart, 2017); liquidity and potential for diversification (Wei, 2018; Platanakis et al., 2018; Dyhrberg et al., 2018; Liu, 2019). Corbet et al. (Corbet et al., 2019) provide a literature review of the topics that have attracted most of the attention lately.

2.2. Neural networks in the financial and cryptocurrency domains

Albeit most of the early studies applying deep learning in the financial domain, are focused on identifying dependencies between stock market movements and news events using Deep Convolutional Neural Networks and Recurrent Neural Networks (RNN) (Ding et al., 2015; Yoshihara et al., 2014), its application to extract information from the stock return time-series is currently growing. In fact, recently (Moews et al., 2019; Adcock & Gradojevic, 2019) show how deep feed-forward neural networks are suitable for learning lagged correlations between the step-wise trends of a large number of financial time-series such as cryptocurrency returns. Many systems based on neural networks for trading strategies in cryptocurrency markets using prices have been proposed (Atsalakis et al., 2019; Vo & Yost-Bremm, 2018; Nakano et al., 2018). Silva de Souza et al. (Silva de Souza et al., 2019) study how strategies based on Artificial Neural Networks among other techniques can generate abnormal risk-adjusted returns when applied to Bitcoin. In their analysis, they defend that strategies based on neural networks may beat buy-and-hold strategies even during strong bull trends as these are able to explore short-run informational inefficiencies and generate abnormal profits. Furthermore, in recent times there are several studies using deep learning algorithms (Zhang et al., 2019; Sirignano & Cont, 2019; Mäkinen et al., 2018) which claim that there might be a universal price formulation for the deterministic part of trading behavior to some degree. This would imply that financial data at high-frequency could have stationary patterns over long time periods that can be learned (Shintate & Pichl, 2019).

In this space, there are a few recent papers applying different architectures for price prediction and trend classification on the short-term, in mid and high frequencies.

The first group of these studied in the literature are recurring neural networks (RNNs), especially Long-Short-Term Memory neural networks (LSTMs).

Lahmiri and Bekiros (Lahmiri & Bekiros, 2019) explore the usage of Long-Short-Term Memory neural networks and Generalized Regression Neural Networks (GRNN) for price prediction in Dash, Ripple, and Bitcoin. The LSTM neural network performs clearly better than GRNN in terms of root mean squared error (obtaining less than the half) at the daily level.

Bao et al. (Bao et al., 2017) propose a deep learning framework over technical indicators, prices, and macroeconomic variables to forecast the next day's closing price. They combine wavelet transforms, stacked auto-encoders, and LSTM neural networks for stock price forecasting in six popular market indexes. Nelson et al. (Nelson et al., 2017) use an LSTM neural network to predict future trends of stock prices. They use a set of technical indicators and the price history in 15-minutes intervals. Borovkova and Tsiamas (Borovkova & Tsiamas, 2018) feed technical indicators to an online ensemble of LSTMs to predict stock price on 5-minute intervals. Their approach is able to deal with non-stationarities in different stock market indexes. Fisher and Kraus (Fischer & Krauss, 2018) compare LSTM neural networks with Random Forest, deep neural networks (DNN), and Logistic Regression for S&P500 price prediction. The LSTM model obtained both the best accuracy and daily returns.

Wu et al. (Wu et al., 2019) propose a framework to select input variables for LSTM models in Bitcoin price forecasting. Kwon et al. (Kwon et al., 2019) use LSTM networks to classify the price trend (price-up or price-down) of different cryptocurrencies obtaining better results than Gradient Boosting Models. Miura et al. (Miura et al., 2019) benchmark LSTMs, MLP, and gated recurrent units on Bitcoin prices to predict realized volatility. On their study LSTMs and gated recurrent units performed better compared to MLPs.

Mallqui and Fernandes (Mallqui & Fernandes, 2019) compare different ensembles and neural networks to classify Bitcoin price trend, closing, maximum and minimum price. In their study RNNs and MLPs obtain the best results for price trend and closing price prediction respectively.

The second group of architectures widely used in the literature are Deep Neural Networks (DNN) such as feed-forward networks, or more specifically, MLPs. Adcock & Gradojevic (Adcock & Gradojevic, 2019) use feed-forward neural networks with lagged returns and simple technical trading rules to forecast BTC/USD returns. The authors conclude that these architectures are suitable for Bitcoin returns forecasting, although the results might vary over time impacted by its sharp price movement. In order to assess this

volatility in Bitcoin prices, Kristjanpoller and Minutolo (Kristjanpoller & Minutolo, 2018) propose a hybrid approach combining lagged values, technical indexes, and econometric models, preprocessed with Principal Components Analysis and fed into an MLP architecture.

Nakano et al. (Nakano et al., 2018) use a DNN to predict price direction on Bitcoin 15-minutes time intervals using prices and technical indicators. Their intraday strategy obtains better returns than buy-and-hold and other primitive technical trading strategies. Chong et al. (Chong et al., 2017) propose a deep feature learning-based stock market prediction model. They use principal component analysis (PCA), an auto-encoder, and a restricted Boltzmann machine, with a three-layer DNN to predict the stock returns at the 5-minutes level in the Korean stock market. Das et al. (Das et al., 2018) examine the predictability of the S&P500 Index using past returns of all the stocks in the index through the use of DNN. They train a feed-forward deep learning network with three hidden layers of 200 nodes each to predict the direction of the closing price from 5 to 30 days ahead. Singh and Srivastava (Singh & Srivastava, 2017) combine PCA with RNN, Radial Basis Function Neural Networks and DNN for trend prediction in NASDAQ daily prices. DNNs outperformed the other neural network architectures in the experiments.

The third group architectures studied are the ones based on principles of the field of image processing, such as Convolutional Neural Networks. Selvin et al. (Selvin et al., 2017) compare LSTMs, RNNs, and CNN architectures using a sliding window approach for short-term future stock price prediction. In their approach, the CNN architecture outperforms LSTM and RNN.

Shintate and Pichl (Shintate & Pichl, 2019) compare LSTMs and MLPs on Bitcoin and Litecoin exchange time-series at 1-minute intervals. They also propose their own algorithm (RSM), namely Random Sampling Method, based on deep learning developments in the field of image processing. RSM obtains the best accuracy when compared to LSTMs and MLP on their study. Hiransha et al. (Hiransha et al., 2018) compare Multilayer Perceptron (MLP), RNN, LSTM and CNN architectures for predicting the stock price of highly traded companies in the National Stock Exchange (NSE) of India and the New York Stock Exchange (NYSE). In their study, the model with best results was the CNN architecture, outperforming as well as other classical models as ARIMA at the day level. Sezer & Ozbayoglu (Sezer & Ozbayoglu, 2018) propose a trading algorithm using a 2D CNN architecture at the daily level to determine buy and sell points in the stocks market. Their model

outperformed Buy & Hold, MLPs and LSTMs on short and long out-of-sample periods in their study. Tsantekidis et al. (Tsantekidis et al., 2017) propose a CNN architecture to predict price trend in stock prices on high frequencies using data from limit order books. Their results show how CNNs beat other state-of-the-art algorithms for the same purpose.

To the best of our knowledge, despite being one of the techniques obtaining best results on financial prediction in the literature, image processing based architectures are the least explored of the above-mentioned groups. On top of it, there is a gap regarding the application of Convolutional Neural Networks for cryptocurrency forecasting and price prediction at high frequencies. CNNs architectures have been the most adopted deep learning model and have become a de facto standard for image classification and computer vision in recent years (Canziani et al., 2016). However, a frequent issue of CNNs is that these tend to ignore the latent dynamics existing in the data. Our proposed methodology encodes a time series as a 2D image-like structure to take into account dependencies from recent market movements and also potential recurrences.

The architectures used in this study are briefly described in the section that follows.

3. Architectures considered

3.1. Multi-layer perceptrons

The Multi-layer Perceptron (MLP) is one of the most popular classic neural network architectures. It is a type of feed-forward neural network which originally consists of at least three layers: an input layer, a hidden layer, and an output layer (Guresen et al., 2011).

Input data is propagated forward to the output neuron, and each intermediate unit is fully connected to the neurons of the next layer through weighted connections (Hiransha et al., 2018). Learning occurs by changing these connection weights, often through a gradient descent-based approach like the back-propagation algorithm, to minimize the error obtained. An example of an MLP architecture can be seen in Figure 1.

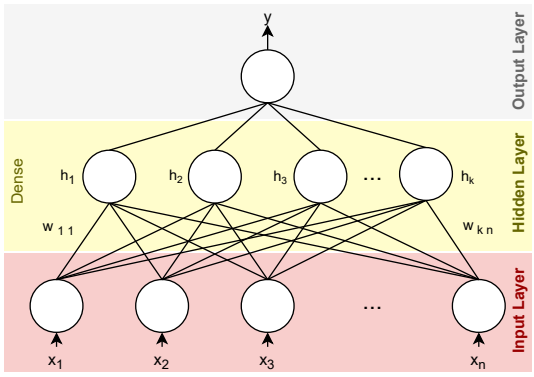


Figure 1: Example of an MLP architecture for a single hidden layer, n neurons in the input and k neurons in the hidden layer. x,y and w represent the input features, the prediction, and the weighted connections respectively.

3.2. Radial Basis Function networks

Radial Basis Function Neural Networks (RBFNNs) are feed-forward neural networks with a similar setting to MLP architectures. As can be seen in the example architecture in Figure 2, the main difference between RBFNNs and MLPs relies on the hidden layer.

First, RBFNNs apply Gaussian activation functions (denoted by φ in Fig. 2) while MLPs use sigmoidal or other monotonic functions as ReLU. Second, RBFNNs compute Euclidean distances between the weights (centers) and the input neurons rather than dot products.

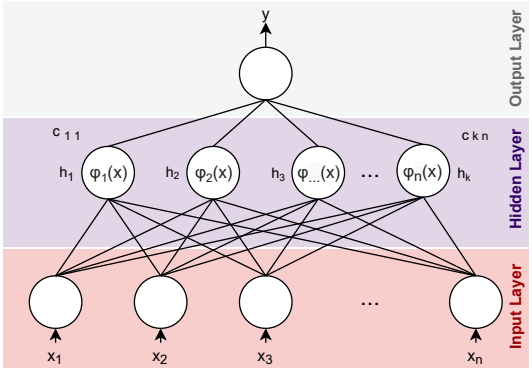


Figure 2: Example of a radial basis function neural network (RBFNN) architecture for a single hidden layer, n neurons in the input and k neurons in the hidden layer. x,y and c represent the input features, the prediction, and the hidden layer centers respectively.

In other words, RBFNN architectures store prototypes in their hidden layer, used to compute the average Euclidean distance as a similarity measure. This allows the classifier to identify when a test example could represent a novel class, or a new regime in the domain time series forecasting (Liu & Zhang, 2010). It is demonstrated that their related cost function is local minima free with respect to all the network weights (Serrano, 2019).

3.3. Convolutional neural networks

Convolutional Neural Networks (CNN) constitute another type of feed-forward architecture which often take their input as bi-dimensional matrices. They typically consist of a set of successive convolutional and subsampling layers, one or more hidden layers and an output layer. The first two types of layers are combined n-times to extract high-level feature vectors in one dimension. These feature vectors are processed by the hidden and output layers, that work like a fully connected multilayer perceptron (Sezer & Ozbayoglu, 2018). An example of a CNN architecture can be seen in Figure 3.

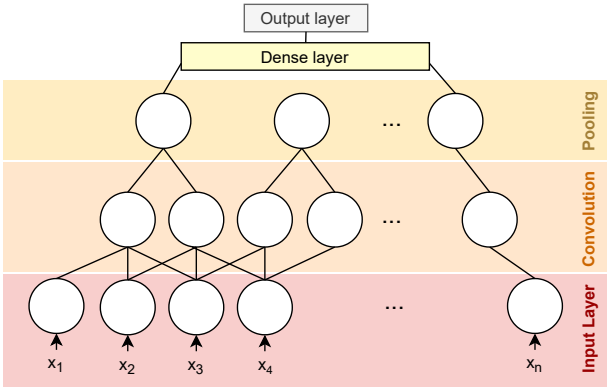


Figure 3: Example of a CNN architecture for a single convolutional layer, a single pooling (subsampling) layer, a single dense (fully-connected) layer and n neurons in the input. x represents the input features flattened.

In this architecture, convolutional layers consist of multiple filters or convolutions that are applied to the input from the previous layer. Convolution filter kernel weights are optimized during the training process. In this context subsampling or pooling layers reduce the dimension of the features, acting as a mechanism of robustness against noise.

3.4. Long Short-Term Memory neural networks

Long Short-Term Memory neural networks (LSTM), proposed by Hochreiter and Schmidhuber (Hochreiter & Schmidhuber, 1997), are designed to keep adjacent temporal information whilst they are able to remember information for a long time in their cells. In this regard, LSTM neural networks provide an extension to recurrent neural network architectures. Albeit LSTM consists of a memory block instead of a neural network layer with a feedback loop as normal RNNs. Each LSTM memory block (or cell) is supported by three components: the input, forget and output controlling gates.

The forget gate retrieves information from the prior state in the last memory block (long-term state) using the previous time-step output (or previous short-term state). It controls which information should be removed. The *input gate* determines the amount of information needed in order to generate the current state. It controls which information is added to the cell (or long-term) state. Finally, *output gates* act as filters and control which information from the current state produces the output (prediction) or short-term state. An example of an LSTM block can be seen in Figure 4.

In this work we are not going to use a simple LSTM network, but instead our configuration will be a hybrid deep network that combines convolutional layers, LSTM layers and dense layers for output, called CLSTM hereafter.

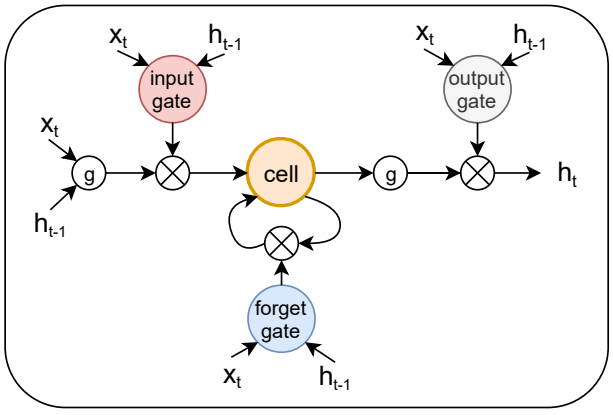


Figure 4: Example of am LSTM block. x and h represent the input and the predicted output (short-term state) respectively. g represents a tangent function.

4. Experimental design

4.1. Sample

The core data set used in the experimental analysis will cover six of the most popular cryptocurrencies (Bitcoin, Dash, Ether, Litecoin, Monero, and Ripple) over a year of data (third quarter of 2018 to second quarter of 2019). The price series vs. USD, sampled at 1-minute intervals over the period from the 1st of July of 2018 to the 30rd of June of 2019, was sourced from the cryptocurrency data provider Cryptocompare.

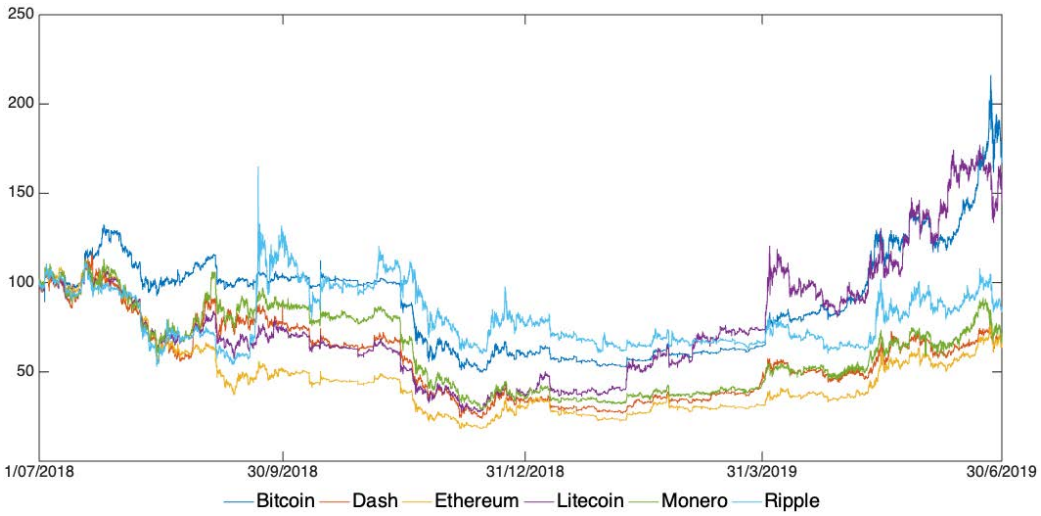


Figure 5: Evolution of exchange rates vs. USD at a 1-minute resolution from Q3 2018 to Q2 2019. Data scaled to base 100.

Figure 5 shows the behavior of the exchange rates vs. USD at the minute bar over the mentioned period. The series were scaled to base 100 for the sake of clarity.

In this study, we will rely on 18 technical indicators based on a popular set first used by Kara et al. (Kara et al., 2011) that we extended with additional long and short moving averages (5, 20, 30 and 60 minutes). The set includes popular momentum-based indicators such as Commodity Channel Index, Momentum, Moving Average Convergence/Divergence, Relative Strength Index, Stochastic, and Williams' R.

This list of technical indicators is a representative set of the trend-following technical indicators covered by the relevant literature (Hsu et al., 2016). It

puts together previous research works that use neural networks in the financial domain (Diler, 2003; Armano et al., 2005; Huang & Tsai, 2009; Kim & Han, 2000; Yao et al., 1999) and overlaps across most of the papers for price trend prediction on the financial domain and cryptocurrencies. (Adcock & Gradojevic, 2019; Nakano et al., 2018; Kristjanpoller & Minutolo, 2018). It has also been used later by other authors (Hsu et al., 2016; Patel et al., 2015).

If we consider how these two families of indicators are used by practitioners, on one hand, moving-averages are often used to define trading rules that generate buy or sell signals based on the relative behavior of indicators calculated over shorter and longer time periods. The comparison of these lagging indicators reveals changes in stock price trends based on the principle that short moving-averages are more sensitive to recent price movements than the long ones. Momentum indicators, on the other hand, measure price differences over relatively short periods of time to track the speed of price changes. These are used by investors to measure the strength of trends and are often used to forecast reversals that are subsequently used to define trading signals. Neely et al. Neely et al. (2014) provide examples of the structure of some of these rules.

In summary, this set of features, which is formally defined in Table 1, consists of the following:

- Accumulation/Distribution Oscillator (A/D)
- Commodity Channel Index (CCI)
- Larry William's R (LWI)
- Momentum
- Moving average convergence divergence (MACD)
- Relative Strength Index (RSI)
- Simple n-second moving average (SMA) over 5, 10, 20, 30 and 60 time periods
- Stochastic D% (SD)
- Stochastic K% (SK)

Table 1: Technical indicators used in the analysis. Formulas as reported in (Kara et al., 2011).

Indicator	Formula
A/D	$\frac{H_t - C_{t-1}}{H_t - L_t}$
CCI	$\frac{M_t - SM_t}{0.015 D_t}$
LWR	$\frac{H_n - C_t}{H_n - L_n} \times 100$
MACD	$MACD(n)_{t-1} + 2/n + 1 \times (DF_t - MACD(n)_{t-1})$
Momentum	$C_t - C_{t-n}$
RSI	$100 - \frac{100}{1 + (\sum_{i=0}^{n-1} Up_{t-i}/n)/(\sum_{i=0}^{n-1} Dw_{t-i}/n)}$
SMA (5,10,20,30,60)	$\frac{C_t + C_{t-1} + ... + C_{t-n+1}}{n}$
SD	$\frac{\sum_{i=0}^{n-1} K_{t-i}\%}{n}$
SK	$\frac{C_t - LL_{t-n}}{HH_{t-n} - LL_{t-n}} \times 100$
WMA (5,10,20,30,60)	$\frac{n \times C_t + (n-1) \times C_{t-1} + ... + C_{t-n+1}}{n + (n-1) + ... + 1}$

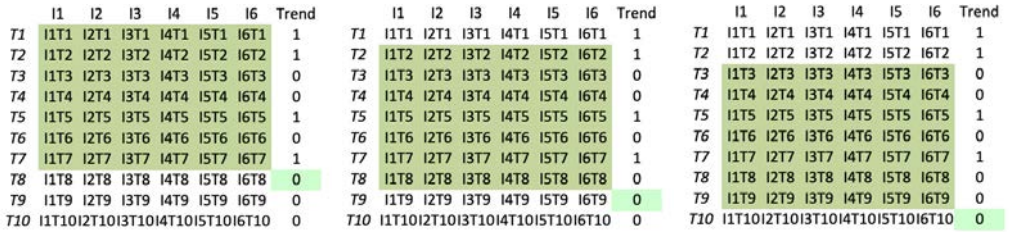
C_t: closing price; L_t: lowest price; H_t: highest price at time t; DF: $EMA(12)_t - EMA(26)_t$; EMA: Exponential moving average; $EMA(k)_t$: $EMA(k)_{t-1} + \alpha \times (c_t - EMA(k)_{t-1})$; α: smoothing factor: $2/1 + k$; k: time period of k minute exponential moving average; LL_t and HH_t: mean lowest low and highest high in the last t minutes; $M_t : H_t + L_t + C_t/3$; $SM_t : \sum_{i=1}^{n} M_{t-i+1})/n$; $D_t : (\sum_{i=1}^{n} |M_{t-i+1} - SM_t|)/n$; Up_t: upward price change; Dw_t: downward price change at time t.

- Weighted n-second moving average (WMA) over 5, 10, 20, 30 and 60 time periods

While it could be argued that the number of indicators could be greater, and others such as volume indicators have not been used to identify trends, the current selection is limited due to the high computational cost of algorithms used. Our setup is in the same range of indicators than other recent papers in the academic literature (Xu et al., 2019; Sun et al., 2019; Hoseinzade & Haratizadeh, 2019; Picasso et al., 2019; Kristjanpoller & Minutolo, 2018).

Given that the indicators require lagged information, we extended the sample slightly with the last minutes of the 30$^{\text{th}}$ of June of 2018 to ensure that we had the 18 required features from the very start of the 1$^{\text{st}}$ of July.

We have constructed our data set, where each pattern contains the values for the i indicators for "l" consecutive lags for each pattern plus the trend (that is, whether the value appreciated or not) as the class to be predicted. This process is shown in Figure 6, on an example that uses 6 indicators and 7 time lags as the window size. The relevant trend to predict is the value



	I1	I2	I3	I4	I5	I6	Trend
T1	I1T1	I2T1	I3T1	I4T1	I5T1	I6T1	1
T2	I1T2	I2T2	I3T2	I4T2	I5T2	I6T2	1
T3	I1T3	I2T3	I3T3	I4T3	I5T3	I6T3	0
T4	I1T4	I2T4	I3T4	I4T4	I5T4	I6T4	0
T5	I1T5	I2T5	I3T5	I4T5	I5T5	I6T5	1
T6	I1T6	I2T6	I3T6	I4T6	I5T6	I6T6	0
T7	I1T7	I2T7	I3T7	I4T7	I5T7	I6T7	1
T8	I1T8	I2T8	I3T8	I4T8	I5T8	I6T8	0
T9	I1T9	I2T9	I3T9	I4T9	I5T9	I6T9	0
T10	I1T10	I2T10	I3T10	I4T10	I5T10	I6T10	0

(a) Pattern 1

	I1	I2	I3	I4	I5	I6	Trend
T1	I1T1	I2T1	I3T1	I4T1	I5T1	I6T1	1
T2	I1T2	I2T2	I3T2	I4T2	I5T2	I6T2	1
T3	I1T3	I2T3	I3T3	I4T3	I5T3	I6T3	0
T4	I1T4	I2T4	I3T4	I4T4	I5T4	I6T4	0
T5	I1T5	I2T5	I3T5	I4T5	I5T5	I6T5	1
T6	I1T6	I2T6	I3T6	I4T6	I5T6	I6T6	0
T7	I1T7	I2T7	I3T7	I4T7	I5T7	I6T7	1
T8	I1T8	I2T8	I3T8	I4T8	I5T8	I6T8	0
T9	I1T9	I2T9	I3T9	I4T9	I5T9	I6T9	0
T10	I1T10	I2T10	I3T10	I4T10	I5T10	I6T10	0

(b) Pattern 2

	I1	I2	I3	I4	I5	I6	Trend
T1	I1T1	I2T1	I3T1	I4T1	I5T1	I6T1	1
T2	I1T2	I2T2	I3T2	I4T2	I5T2	I6T2	1
T3	I1T3	I2T3	I3T3	I4T3	I5T3	I6T3	0
T4	I1T4	I2T4	I3T4	I4T4	I5T4	I6T4	0
T5	I1T5	I2T5	I3T5	I4T5	I5T5	I6T5	1
T6	I1T6	I2T6	I3T6	I4T6	I5T6	I6T6	0
T7	I1T7	I2T7	I3T7	I4T7	I5T7	I6T7	1
T8	I1T8	I2T8	I3T8	I4T8	I5T8	I6T8	0
T9	I1T9	I2T9	I3T9	I4T9	I5T9	I6T9	0
T10	I1T10	I2T10	I3T10	I4T10	I5T10	I6T10	0

(c) Pattern 3

Figure 6: Pattern generation process. Example based on 6 technical indicators and 7 lags. Class is highlighted in the "Trend" column. Matrix format is used for convolutional neural networks, MLP and RBFNN use a flattened 1-dimensional vector.)

corresponding to the following period of time, shown in the last column. Therefore, our system predicts the outcome for a given time step $(t + 1)$ using the indicators that correspond to the previous l time steps (t, $t - 1$, $\ldots$, $t - l + 1$). For the intermediate time steps, we don't use the trend column. For the MLP and RBFNN approaches, patterns are converted in one-dimensional vectors of $l \times i + 1$ attributes, including the class. For the CNN and CLSTM network approach, patterns are $l \times i$ matrices with an additional class value per pattern. Note that the process of finding a good value for l, is part of the experimental protocol.

In Convolutional layers, neurons only receive input from a subarea of the previous layer (Dresp-Langley et al., 2019). The proposed CNN architecture aims to learn the relevant technical indicators during the training process by extracting feature maps that represent price trends. The relationships between these price trends are later fed into a fully connected layer to learn the different relationships present. This automates the behavior of chartists and technical analysts that often make decisions based on the current and previous values of several technical indicators, trendlines and their price patterns such as triangles, bears, flags, etc. (Campbell et al., 1997).

In order to avoid temporal bias in the data generation process, we randomize the order of the aforementioned patterns. That is, our hypothesis is that our system will recognize correlations between values of the l time steps regardless of their occurrence in the complete interval analyzed.

The application of the mentioned approach on the data set described above resulted in the generation of 525,600 patterns to be split into three samples. The first one consisted of 70% (367,920 patterns) used for training. The second one included 15% (78,840 patterns) and served as the validation

set. Finally, the remaining 15% was reserved for testing purposes. Table 2 reports the breakdown of pattern classes for the seven cryptocurrencies by sample. As it was mentioned before, the class *1* represents appreciation vs USD in the relevant one-minute interval, and *0* is used for the rest.

Table 2: Breakdown of pattern classes for the six cryptocurrencies by sample. Patterns computed at 1-minute intervals for the period Q3 2018 to Q2 2019. The values for the test case are used later as baseline classifier accuracy for comparison.

		Bitcoin	Dash	Ether	Litecoin	Monero	Ripple
Train							
	0	49.86%	70.95%	54.68%	60.47%	74.44%	62.81%
	1	50.14%	29.05%	45.32%	38.53%	25.56%	37.19%
Validation							
	0	49.64%	70.79%	54.55%	60.65%	74.45%	62.74%
	1	50.36%	29.21%	44.45%	39.35%	25.55%	37.36%
Test							
Baseline	*0*	50.00%	70.62%	54.67%	60.88%	74.69%	62.81%
	1	50.00%	29.38%	45.33%	39.12%	25.31%	37.19%

4.2. Experimental protocol

In order to perform a fair comparison of the four types of models, we shall perform several stages of preliminary testing before selecting the best configuration of each model for the final experimental round.

The first stage is to decide the values for some parameters that affected the construction of the data sets themselves. This includes the parameters of the technical indicators, and also the number of lags in each pattern (parameter l). After these values are selected, we construct the data files that contain train, validation and test data and use them for all the subsequent experiments. For this stage, the networks will be trained using the train set and results will be compared using the validation set.

The second stage entails identifying appropriate network structures and learning hyperparameters for each of the models. To this end, we will run exploratory experiments and the best configuration will be selected for each model, again using the result on the validation set.

For MLP we shall test several combinations of number of hidden layers, number of neurons in each layer and learning rates. For RBFNN we shall

try several values of the number of neurons and the β parameter. For each combination of configuration and cryptocurrency we will run three experiments. In the final stage we shall use the best performing configurations on validation test for each cryptocurrency. That means that we shall keep at most 6 different configurations for MLP and 6 for RBFNN.

For CNN, because of the much higher computational cost of training and testing, at this stage we shall explore five different architectures, with four values for the Learning rate, but tests will be performed on a single cryptocurrency, Bitcoin. According to Goodfellow et al. (Goodfellow et al., 2016), these two aspects might be the two most important parameters for CNN networks. The best configuration will then be used for all cryptocurrencies.

Finally, for the CLSTM network we shall use a configuration based on (Stoye, 2018), which we think is complex enough to adapt to the current application.

The third and final stage will compare the performance of all four neural network types across the six cryptocurrencies. In this case we shall use the models of each type with the best configurations selected as defined above, but results will be reported on the the accuracy on the test set data (instead of the validation set) that had been not been used in the previous stages. This separate test is required to ensure the validity of the comparison.

For statistical validation of the results, in stage 3 we shall perform a series of 20 experiments (train and test) for each configuration. Given the stochastic component introduced in the initialization of weights, the statistical significance for the average performance differences will be formally tested. First, we will start testing the normality of the distribution of accuracy with a Kolmogorov-Smirnov test, using the Lilliefors correction. In case the tests of normality were rejected, we would apply a non-parametric test, Wilcoxon's sign ranges. Otherwise, we would test for homoskedasticity using Levene test and, depending on the result, we would test for equality of means either using a Welch test, or a standard t-test.

In addition to that, we will test the statistical significance of the relative predictive performance of the median models on test sample by cryptocurrency using the Giacomini and White test. Given that data snooping might hinder the validity of the results, we will rely on White's reality check for data snooping to safeguard it.

4.3. Parametrization

The computation for the technical indicators relies on a number of periods, n, that was set to 10 minutes. This parameter was defined at the start

and not optimized. For the two indicators that are moving averages, we consider an extended set of periods of 5, 10, 20, 30 and 60 minutes. The features were computed with the technical analysis library TA-lib[1] and we used the default values for all parameters other than the mentioned time period.

As we discussed in section 4.2, we performed some preliminary experiments on the combination of the training and validation samples to determine the value of the window size (or the number of lags) l. To that end as initial exploratory values, we performed tests with $l = 15$ and $l = 60$ as the window size. Results are reported in Table A.17. We found an inverse relationship between the value of this parameter and performance and, therefore, we set $l = 15$. This lag was used for all the experiments thereafter.

At that point we proceeded with the exploratory experiments for the MLP and RBFNN on the training and validation samples (the test set was left untouched) whose results can be found in Appendix A. These tables report the sensitivity of these two kinds of neural networks to different combinations of configuration parameters. In the case of MLP, for each of the six cryptocurrencies we validated 60 combinations of learning rate, number of hidden layers and number of neurons per layer (see Tables A.10, A.11, A.12, A.13, A.14 and A.15). For RBFNN, for each cryptocurrency we validated the result of 42 combinations of the number of neurons and values of the width parameter β (see Table A.16). In all cases we calculated the average accuracy on the validation test over three independent experiments. The result of this process was the selection of several sets of best parameters, reported in Table 3.

Regarding CNNs, we also performed an exploratory analysis based on the train and validation samples, but only for the Bitcoin cryptocurrency. The performance of the different configurations is reported in Table A.18 (Appendix A), which shows the classification results for five different CNN architectures, each of them trained using four learning rate values: 0.1, 0.01, 0.001, and 0.0001. The CNN architectures tested in this stage were the following:

- Vertical Filters: it is composed of five similar blocks, each consisting of a convolutional layer, together with batch normalization and a ReLU activation function. The first four blocks also include dropout regularization, while a global average pooling layer follows the fifth block to

[1]Technical Analysis library—http://ta-lib.org/

Table 3: Parametrization for the multi-layer perceptrons (MLP) and the radial basis function neural networks (RBFNN) experiments by cryptocurrency. Bitcoin (BTC), Dash (DASH), Ether (ETH), Litecoin (LTC), Monero (XMR), Ripple (XRP).

		BTC	DASH	ETH	LTC	XMR	XRP
MLP	Transfer funct.	ReLU					
	Epochs (max.)	100					
	Hiden layers	1	1	1	1	1	1
	Neur. hid. layer	10	10	10	12	14	10
	Leaning rate	0.1	0.1	0.1	0.01	0.01	0.1
RBFNN	Initialization	K-means					
	Learning rate	0.001					
	Neurons	80	80	20	100	100	80
	β	2.5	2.5	1	2.5	1.5	3

minimize overfitting by reducing the total number of parameters in the model (Lin et al., 2013). The last layer of the network consists of a single neuron that performs the final prediction after applying a sigmoid activation function. We would like to highlight that all the filters from the convolutional layers have a vertical shape (their shape is Kx1), allowing each kernel to work on a single indicator over time, and thus preventing to perform convolutions that merge multiple indicators. A graphical description of this configuration is shown in Figure 7.

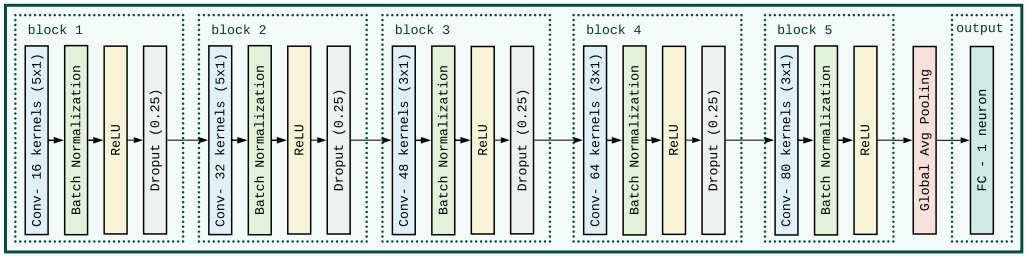


Figure 7: Convolutional neural network architecture "Vertical Filters" : with five Convolutional layers, monotonic activation functions, batch normalization, average pooling and dropout. This configuration was selected as our best CNN architecture for the problem.

- Horizontal Filters: It is composed of four similar blocks, each consisting of a convolutional layer, together with batch normalization and a

ReLU activation function. The first three blocks also include dropout regularization, while a global average pooling layer follows the fourth block as in the previous network. The last layer of the network consists of a single neuron that performs the final prediction after applying a sigmoid activation function. We would like to highlight that all the filters from the convolutional layers have a horizontal shape (their shape is 1xK), allowing each kernel to work on a single time instant over multiple indicators, and thus preventing to perform convolutions that merge multiple time instants.

- DNN: a custom deep neural network that consists of three convolutional layers (with 7x7 filters, 3x3 filters, and 3x3 filters, respectively), and a 1000-neuron fully connected layer prior to the last layer of the network (single neuron).

We also tested two predefined network architectures submitted to the ImageNet Large Scale Visual Recognition Challenge 2015 (ILSVRC'15):

- Xception (Chollet, 2016) (by Google): even better results than Inception-v3 (Szegedy et al., 2015), which was the first runner up network in ILSVRC'15.

- ResNet50 (He et al., 2015) (by Microsoft): 50-layer version of the network that won ILSVRC'15.

As can be seen in Table A.18 (Appendix A), Vertical Filters obtained the best results in our tests compared to all the other architectures described above. This performs especially better when using a Learning rate of 0.001. As a consequence, this was the architecture used for the final stage (Figure 7).

The Convolutional LSTM architecture selected for the final stage is based on the one used in (Stoye, 2018). We performed some small modifications to help reducing the computational cost of training and testing: our version of this architecture has a first section with five convolutional with 1x1 filters; the output of these layers is a vector of learned features with the same dimension as the initial input; then, these features are fed to a single LSTM layer with 150 units. Finally, output is processed by a complex multilayer perceptron with 8 layers, where the first one has 200 neurons and the rest 100 (Figure 8).

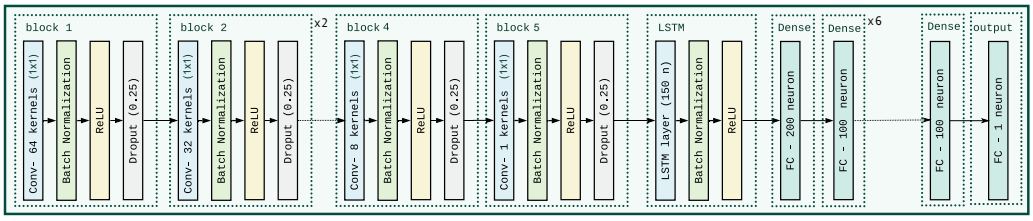


Figure 8: Selected CLSTM neural network architecture with five Convolutional layers plus a single LSTM layer and 8 fully connected MLP as output (*Dense* layers) for output.

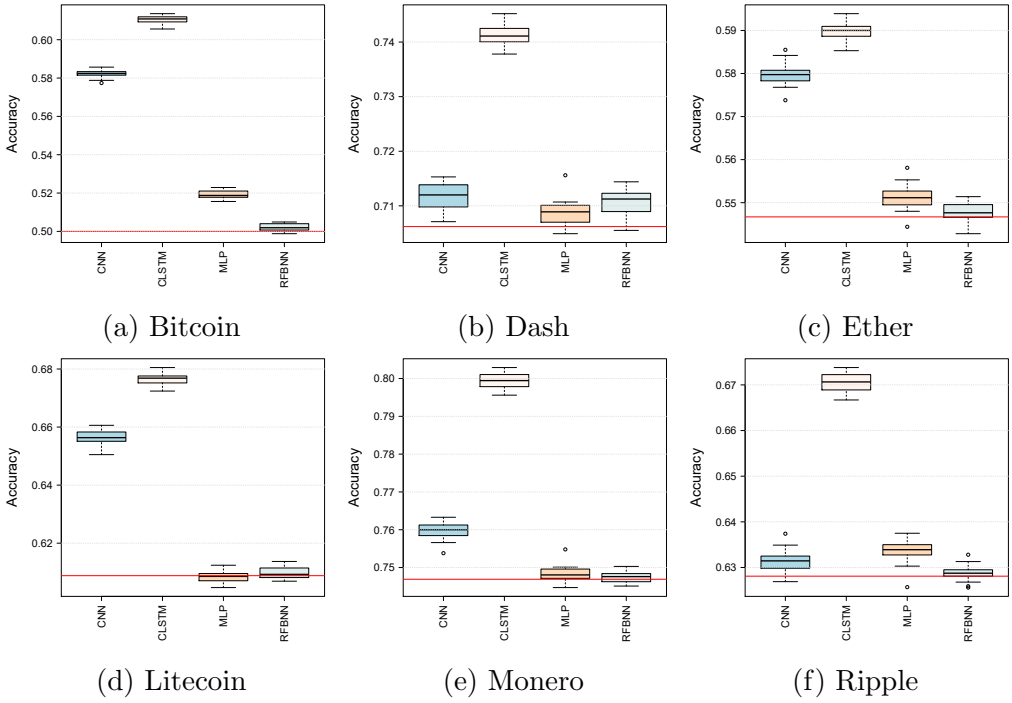
5. Experimental results and discussion

In this section we report the outcome of the evaluation of the four approaches on the reserved test set using the parameterizations identified in the exploratory analysis.

The main experimental results are illustrated in Fig. 9. There, we can see boxplots of accuracy on test sample by cryptocurrency and network structure. Results are based on 20 runs of the experiments. In Table 4 we report the numeric results for all these experiments. In the boxplots we show as an horizontal line the baseline result that corresponds to a trivial classifier that always predicted the most frequent class for each currency. These baselines are reported in Table 2.

Results clearly show that the CLSTM architecture clearly outperforms the rest for all of the cryptocurrencies. CNN achieves comparable results in Bitcoin, Ether and Litecoin, and also Monero to certain extent. Though results for RFBNN and MLP are poor in general, MLP seems to be able to achieve average results slightly above the baseline in Bitcoin, Dash, Ether and Ripple (where it outperforms CNN slightly). RFBNN shows poor results in all but the Dash currency.

In order to provide more soundness to the study, we formally tested the statistical significance of the relative predictive performance of the median models on test sample by cryptocurrency. To that end, we used both the Diebold and Mariano (Diebold & Mariano, 1995) and Giacomini and White (Giacomini & White, 2006) tests. These evaluate the null hypothesis that the two forecasts have the same accuracy (the metric of the models is different or not). The p-values, reported in Tables 5 and 6, respectively, show the results as set of six double-entry tables that compare the performance of the neural network architectures in rows with the ones in columns.



(a) Bitcoin (b) Dash (c) Ether

(d) Litecoin (e) Monero (f) Ripple

Figure 9: Boxplots of accuracy on test data sets. Convolutional neural network (CNN), hybrid CNN-LSTM network (CLSTM), multilayer perceptron (MLP) and radial basis function neural network (RBFNN). Results over 20 experiments. Red lines show the baseline accuracy of a trivial most frequent class classifier.

Something that is apparent when we look at Tables 5 and 6 is that, generally speaking, Bitcoin seems to be more predictable than the rest. In order to gain a better understanding of this possibility, we assess the predictability of each cryptocurrency in Table 8, where we show the amount by which the mean of both methods or the best method are able to improve over the baseline accuracy.

Once again the strongest indication of predictability is obtained for Bitcoin, followed by Ether, while results for the other three Bitcoins show less than a 1% improvement.

Something to consider in the previous analysis is the potential presence of data mining biases. More specifically, the risk of data snooping introduced by repeated testing of models on the same data set. In order to safeguard the validity of the results, we relied on the Model Confidence Set procedure defined by Hansen et al. (Hansen et al., 2011) as implemented in R

by Bernardi & Catania (Bernardi & Catania, 2018). Given a collection of models, this generalization of White's reality check (White, 2000) discards sequentially the worst-performing ones until the null hypothesis of equal predictive ability cannot be rejected. Once the elimination process is completed, the remaining subset contains the best models at a given confidence level.

In order to obtain more informative results, we added two models to the initial collection of artificial neural networks. The first one was a series of random predictions matching the distributions observed in train and validation samples. The second naive prediction method used as prediction for the whole test set the most frequent class in train and validation samples.

Table 9 reports the model elimination order on test sample by cryptocurrency based on 1000 resamples and a significance level $\alpha = 0.05$. As we can see, CLSTM is identified across cryptocurrencies as the only component of the Superior Set Model. In all six cases, the p-values were well below 0.01.

6. Limitations

This study on the suitability of artificial neural networks to predict trends in high frequency cryptocurrency data using technical indicators adds new relevant evidence on the importance of convolution and deep learning in this domain. However, even though the results of the experiments are promising, it is important to point out that there are several reasons why trend prediction efforts, like the ones described in this study, might not necessarily translate into profits. Among them, we could mention response times, large trends, liquidity issues or transaction costs.

Concerning response time, it is worth noting that all the mentioned algorithms are able to predict the trend in reasonable times considering the data 1-minute time step. More specifically, RBFNN, MLP, CNN and CLSTM required 0.034, 0.279, 0.757 and 1.967 seconds, respectively, on a 16GB NVIDIA Tesla V100 GPU, leaving, in the worst-case scenario, 58 seconds. However, besides the time required for prediction, there are other factors to be considered. For instance, we require an online calculation of the indicators, a process that was executed as a batch task in our test environment. In an operational environment, indicator computation should be streamlined. In addition to this, we would have to consider some additional internal factors, like the time consumed by the system that feeds on the trend predictions to generate the trading signals, and external ones such as propagation time for orders, that depends on the structure of the market considered.

Exchange rates are dominated by larger trends. Given the frequency chosen and the computational cost of the techniques used in the study, the period considered was limited to one year. This means that good results might not be guaranteed for other periods.

There are problems of a different nature. Liquidity, for instance, is known to differ widely among cryptocurrencies. Wei (Wei, 2018) recently analysed 456 of them and, while the main ones like Bitcoin and Ether are very liquid, there are many others that are very thinly traded. This poses a challenge to exploit profitably trend predictions. Regarding transaction costs. Even though the one associated with Bitcoin is lower than that of retail foreign exchange markets Kim (2017), that is not the case for all cryptocurrencies . These costs should be factored into the trading strategy built on top of the trend predictions to make sure that these costs do not end up eroding completely the gains that might be derived from the ability to predict, to a certain extent, the direction of short-term market movements.

Our proposal makes no assumption on whether these issues can be effectively solved. Overall, cryptocurrency trading should operate on the assumption of fast links, low transaction costs and high liquidity, assumptions that may not hold in particular situations.

7. Summary and conclusions

The purpose of this work is to explore the suitability of convolutional neural networks with convolutional components to make intraday trend classification for cryptocurrency exchange rates vs USD based on technical indicators, and studying whether these models add value over traditional alternatives like multilayer perceptrons or radial basis function neural networks. In addition to that, we analyze whether some cryptocurrencies are more predictable than others using the mentioned approaches.

The analysis was based on 1-minute exchange rates over an entire year period (July 1st, 2018, to June 30rd, 2019), and the problem was defined as the prediction of the trend (increase or neutral/decrease) for a given time step by using indicator information on a predefined number of time steps. Data was captured and processed for six of the most popular cryptocurrencies: Bitcoin, Dash, Ether, Litecoin, Monero, and Ripple.

Our work was performed in two phases: first, a preliminary study was able to identify appropriate values for four different types of experimental variables: indicator setup, pattern lag, network architecture, and network

training hyperparameters. The study considers four types of network: Convolutional neural network, hybrid Convolutional LSTM, radial basis function neural network, and conventional multilayer perceptron.

Secondly, in order to assess whether the former classifiers are suited for trend prediction on cryptocurrencies, these architectures were tested for each of the six cases.

Our experimental results support the conclusion that CNNs and specially Convolutional LSTM are suitable as predictors for the price trend when using the defined indicators and parameters on most cryptocurrencies. This was especially true for Bitcoin, Ether and Litecoin. Results of the CLSTM architecture were always significantly better than the rest, and this network was the only one that could predict the trends of Dash and Ripple with some margin (about 4%) over the trivial classifier.

The performance of the rest of the models was quite limited for Dash and Ripple. This can be explained by the fact that those series may have intrinsically more noise, but it may also be a result of a different temporal behavior that our data generation process failed to capture. There is still a wide range of possibilities to improve these results: we may select more specific data generation parameters for these data sets, with different values for the time period of indicators, window size and/or network structure.

Even though the results show that the two network architectures with convolutional components, specially CLSTM, can be useful predictors of market trends at high frequencies, it is worth noting that the study relies on a limited number of technical indicators. Increasing their range would probably reveal that this performance can be improved and, therefore, our results might be considered a lower-bound. For this reason, future lines of research should consider exploring this possibility.

Despite the promising performance of the deep neural network architectures, the study is focused on short-term trend prediction and is subject to limitations that should be addressed to operate in production in a trading setting. The practical application would require the implementation of a decision component and control for additional aspects like time management, market liquidity, or transaction costs.

Further research may also take into consideration the use of higher frequency data (higher resolutions than 1-min intervals). We also plan on performing a comparison between our approach, based on technical indicators, and an approach based on raw prices; studying dynamic trading strategies vs the current analysis based on single point static predictions, or extending the

number of classes to identify sizable price movements that could potentially cover transaction costs and, therefore, could be interpreted as buy or sell signals.

8. Acknowledgements

We would like to thank the editor and external reviewers for their thoughtful and detailed comments on our paper. We would also like to acknowledge the financial support of the Spanish Ministry of Science, Innovation and Universities under grant PGC2018-096849-B-I00 (MCFin).

Appendix A. Exploratory parameterization tests

The tables on this appendix report the results of the exploratory analysis on train and validation sample. For MLP, every combination of four learning rates, three different number of hidden layers and five different numbers of hidden neurons per layer. We show the average accuracy over three experiments. We provide one table per cryptocurrency. For RBFNN we tested seven different numbers of neurons (20 to 300) with six different values for the β parameter (1.0 to 3.5), and we also show the average result of three independent runs on the validation sample. Finally, for CNN we show the results of the tests performed with different number of lags for all the currencies, and experiments with five different architectures with four different learning rates.

Table 4: Descriptive statistics of prediction accuracy by cryptocurrency and network architecture. Boxplots of accuracy on test data sets. Convolutional neural network (CNN), hybrid CNN-LSTM network (CLSTM), Multilayer Perceptron (MLP) and Radial Basis Function Neural Network (RBFNN). Test results based on 20 experiments.

		Mean	Median	Var.	Max.	Min.
Bitcoin	CNN	0.5822	0.5824	< 0.001	0.5857	0.5774
	CLSTM	0.6106	0.6109	< 0.001	0.6136	0.6056
	MLP	0.5192	0.5188	< 0.001	0.5229	0.5156
	RFBNN	0.5021	0.5019	< 0.001	0.5049	0.4988
	Baseline	0.5000				
Dash	CNN	0.7116	0.7120	< 0.001	0.7153	0.7071
	CLSTM	0.7412	0.7411	< 0.001	0.7452	0.7378
	MLP	0.7088	0.7089	< 0.001	0.7156	0.7049
	RFBNN	0.7106	0.7112	< 0.001	0.7144	0.7055
	Baseline	0.7062				
Ether	CNN	0.5797	0.5797	< 0.001	0.5855	0.5738
	CLSTM	0.5899	0.5900	< 0.001	0.5939	0.5853
	MLP	0.5512	0.5512	< 0.001	0.5581	0.5444
	RFBNN	0.5478	0.5476	< 0.001	0.5514	0.5428
	Baseline	0.5467				
Litecoin	CNN	0.6565	0.6563	< 0.001	0.6606	0.6505
	CLSTM	0.6763	0.6768	< 0.001	0.6805	0.6724
	MLP	0.6084	0.6086	< 0.001	0.6124	0.6047
	RFBNN	0.6097	0.6093	< 0.001	0.6137	0.6069
	Baseline	0.6088				
Monero	CNN	0.7597	0.7600	< 0.001	0.7633	0.7538
	CLSTM	0.7994	0.7994	< 0.001	0.8029	0.7956
	MLP	0.7483	0.7480	< 0.001	0.7548	0.7447
	RFBNN	0.7474	0.7476	< 0.001	0.7503	0.7451
	Baseline	0.7469				
Ripple	CNN	0.6313	0.6314	< 0.001	0.6374	0.6269
	CLSTM	0.6704	0.6706	< 0.001	0.6738	0.6667
	MLP	0.6335	0.6339	< 0.001	0.6375	0.6257
	RFBNN	0.6288	0.6288	< 0.001	0.6328	0.6256
	Baseline	0.6281				

Table 5: Statistical significance of the differences in predictive performance of median models on test sample by cryptocurrency. P-values computed using the Diebold and Mariano test. Convolutional neural network (CNN), hybrid CNN-LSTM network (CLSTM), multilayer perceptron (MLP) and radial basis function neural network (RBFNN).

	Bitcoin			Dash		
	CNN	CLSTM	MLP	CNN	CLSTM	MLP
CLSTM	<0.01			<0.01		
MLP	<0.01	<0.01		0.777	<0.01	
RBFNN	<0.01	<0.01	<0.01	0.087	<0.01	0.047
	Ether			Litecoin		
CLSTM	<0.01			<0.01		
MLP	<0.01	<0.01		<0.01	<0.01	
RBFNN	<0.01	<0.01	0.871	<0.01	<0.01	0.192
	Monero			Ripple		
CLSTM	<0.01			<0.01		
MLP	<0.01	<0.01		0.272	<0.01	
RBFNN	<0.01	0.024	<0.01	<0.01	<0.01	<0.01

Table 6: Statistical significance of the differences in predictive performance of median models on test sample by cryptocurrency. P-values computed using the Giacomini and White test. Convolutional neural network (CNN), hybrid CNN-LSTM network (CLSTM), multilayer perceptron (MLP) and radial basis function neural network (RBFNN).

	Bitcoin			Dash		
	CNN	CLSTM	MLP	CNN	CLSTM	MLP
CLSTM	<0.01			<0.01		
MLP	<0.01	<0.01		0.307	<0.01	
RBFNN	<0.01	<0.01	<0.01	0.035	<0.01	0.134
	Ether			Litecoin		
CLSTM	<0.01			<0.01		
MLP	<0.01	<0.01		<0.01	<0.01	
RBFNN	<0.01	<0.01	0.277	<0.01	<0.01	0.016
	Monero			Ripple		
CLSTM	<0.01			<0.01		
MLP	<0.01	<0.01		0.073	<0.01	
RBFNN	<0.01	0.079	<0.01	0.014	<0.01	<0.01

Table 7: Statistical significance of the differences in predictive performance of median models vs. benchmarks on test sample by cryptocurrency. P-values computed using the White reality check for data snooping with 500 simulations. Benchmark 1: random predictions following the distributions in train and validation sample. Benchmark 2: use as prediction the most frequent class in train sample. Bitcoin (BTC), Dash (DASH), Ether (ETH), Litecoin (LTC), Monero (XMR), Ripple (XRP). Convolutional neural network (CNN), hybrid CNN-LSTM network (CLSTM), multilayer perceptron (MLP) and radial basis function neural network (RBFNN).

	CNN		CLSTM		MLP		RBFNN	
	Base 1	Base 2	Base 1	Base 2	Base 1	Base 2	Base 1	Base 2
BTC	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	0.948	0.444
DASH	<0.01	0.066	<0.01	<0.01	<0.01	0.040	<0.01	0.830
ETH	<0.01	<0.01	<0.01	<0.01	<0.01	0.708	<0.01	0.566
LTC	<0.01	<0.01	<0.01	<0.01	<0.01	0.934	<0.01	1.000
XMR	<0.01	<0.01	<0.01	<0.01	<0.01	0.404	<0.01	0.906
XRP	<0.01	0.084	<0.01	<0.01	<0.01	0.002	<0.01	1.000

Table 8: Average excess prediction accuracy vs. most prevalent class by cryptocurrency. Test results based on 20 experiments. Mean result considers the four architectures. Max. corresponds to the best performing of the four methods.

	Bitcoin	Dash	Ether	Litecoin	Monero	Ripple
Mean	5.354%	1.187%	2.046%	2.894%	1.681%	1.292%
Best	11.064%	3.504%	4.324%	6.754%	5.254%	4.234%

All figures are statistically different from 0 at 1%.

Table 9: Model elimination order according to the Model Confidence Set procedure defined Hansen et.al.. Test results by cryptocurrency based on 1000 resamples and a significance level $\alpha = 0.05$. Base 1: random predictions following the distributions in train and validation sample. Base 2: use as prediction the most frequent class in train and validation sample. Convolutional neural network (CNN), hybrid CNN-LSTM network (CLSTM), multilayer perceptron (MLP) and radial basis function neural network (RBFNN).

	Base 1	Base 2	CNN	CLSTM	MLP	RBFNN
Bitcoin	3	2	5		4	1
Dash	1	3	4		5	2
Ether	1	4	5		2	3
Litecoin	1	3	4		5	2
Monero	1	3	5		4	2
Ripple	1	3	4		5	2

Table A.10: Sensitivity of multilayer perceptron (MLP) to the number of neurons in the hidden layer and the choice of the learning rate for the backpropagation training algorithm. Average over three runs on Bitcoin.

Hidden Layers	Learning Rate	Neurons Hidden Layers 6	8	10	12	14
1	0.1	0.5120	0.5106	0.5132	0.5141	0.5124
	0.01	0.5120	0.5132	0.5074	0.5132	0.5097
	0.001	0.5208	0.5162	0.5217	0.5179	0.5213
	0.0001	0.5071	0.5121	0.5085	0.5089	0.5077
2	0.1	0.5040	0.5015	0.5009	0.4980	0.5011
	0.01	0.4996	0.5043	0.5011	0.5003	0.4993
	0.001	0.4952	0.4996	0.5012	0.5000	0.4991
	0.0001	0.5048	0.5096	0.5105	0.5124	0.5102
3	0.1	0.5016	0.5037	0.5047	0.5051	0.5072
	0.01	0.5074	0.5083	0.4984	0.5036	0.5043
	0.001	0.5064	0.5053	0.5045	0.5025	0.5057
	0.0001	0.5156	0.5163	0.5167	0.5156	0.5130

Table A.11: Sensitivity of multilayer perceptron (MLP) to the number of neurons in the hidden layer and the choice of the learning rate for the backpropagation training algorithm. Average over three runs on Dash.

Hidden Layers	Learning Rate	Neurons Hidden Layers 6	8	10	12	14
1	0.1	0.6999	0.6987	0.6999	0.7030	0.6974
	0.01	0.7014	0.7028	0.7016	0.7046	0.7044
	0.001	0.6996	0.6987	0.6961	0.7010	0.6989
	0.0001	0.7097	0.7090	0.7080	0.7110	0.7119
2	0.1	0.6834	0.6889	0.6917	0.6904	0.6889
	0.01	0.6926	0.6900	0.6933	0.6916	0.6915
	0.001	0.6925	0.6919	0.6968	0.6874	0.6919
	0.0001	0.7068	0.7042	0.7014	0.7016	0.7008
3	0.1	0.6881	0.6894	0.6899	0.6911	0.6920
	0.01	0.6891	0.6866	0.6943	0.6886	0.6843
	0.001	0.7006	0.6986	0.6972	0.6982	0.6974
	0.0001	0.6841	0.6851	0.6847	0.6845	0.6872

Table A.12: Sensitivity of multilayer perceptron (MLP) to the number of neurons in the hidden layer and the choice of the learning rate for the backpropagation training algorithm. Average over three runs on Ether.

Hidden Layers	Learning Rate	Neurons Hidden Layers 6	8	10	12	14
1	0.1	0.5371	0.5429	0.5407	0.5442	0.5383
	0.01	0.5537	0.5549	0.5494	0.5517	0.5516
	0.001	0.5378	0.5421	0.5401	0.5434	0.5412
	0.0001	0.5461	0.5387	0.5396	0.5427	0.5449
2	0.1	0.5276	0.5320	0.5300	0.5293	0.5257
	0.01	0.5308	0.5303	0.5294	0.5291	0.5317
	0.001	0.5418	0.5377	0.5391	0.5409	0.5410
	0.0001	0.5325	0.5314	0.5320	0.5310	0.5318
3	0.1	0.5274	0.5284	0.5273	0.5314	0.5301
	0.01	0.5309	0.5318	0.5354	0.5337	0.5280
	0.001	0.5417	0.5437	0.5406	0.5405	0.5361
	0.0001	0.5316	0.5311	0.5321	0.5289	0.5311

Table A.13: Sensitivity of multilayer perceptron (MLP) to the number of neurons in the hidden layer and the choice of the learning rate for the backpropagation training algorithm. Average over three runs on Litecoin.

Hidden Layers	Learning Rate	Neurons Hidden Layers 6	8	10	12	14
1	0.1	0.5978	0.6014	0.6023	0.6013	0.6032
	0.01	0.6096	0.6111	0.6103	0.6107	0.6082
	0.001	0.5980	0.5941	0.5970	0.5956	0.6002
	0.0001	0.5998	0.6063	0.6059	0.5983	0.5987
2	0.1	0.5898	0.5930	0.5955	0.5890	0.5941
	0.01	0.5917	0.5943	0.5955	0.5932	0.5917
	0.001	0.6048	0.6039	0.6041	0.6044	0.6022
	0.0001	0.5944	0.5964	0.5949	0.5939	0.5952
3	0.1	0.5928	0.5911	0.5937	0.5944	0.5931
	0.01	0.5908	0.5932	0.5934	0.5948	0.5939
	0.001	0.5911	0.5920	0.5939	0.5938	0.5934
	0.0001	0.6065	0.6046	0.6080	0.6047	0.6078

Table A.14: Sensitivity of multilayer perceptron (MLP) to the number of neurons in the hidden layer and the choice of the learning rate for the backpropagation training algorithm. Average over three runs on Monero.

Hidden Layers	Learning Rate	Neurons Hidden Layers 6	8	10	12	14
1	0.1	0.7394	0.7395	0.7349	0.7370	0.7438
	0.01	0.7373	0.7383	0.7405	0.7392	0.7395
	0.001	0.7450	0.7464	0.7477	0.7444	0.7439
	0.0001	0.7363	0.7362	0.7362	0.7390	0.7392
2	0.1	0.7283	0.7290	0.7297	0.7313	0.7292
	0.01	0.7384	0.7410	0.7406	0.7427	0.7398
	0.001	0.7301	0.7308	0.7302	0.7267	0.7284
	0.0001	0.7266	0.7203	0.7260	0.7325	0.7271
3	0.1	0.7304	0.7337	0.7284	0.7275	0.7364
	0.01	0.7380	0.7399	0.7379	0.7358	0.7379
	0.001	0.7311	0.7290	0.7234	0.7279	0.7294
	0.0001	0.7278	0.7269	0.7265	0.7269	0.7243

Table A.15: Sensitivity of multilayer perceptron (MLP) to the number of neurons in the hidden layer and the choice of the learning rate for the backpropagation training algorithm. Average over three runs on Ripple.

Hidden Layers	Learning Rate	Neurons Hidden Layers 6	8	10	12	14
1	0.1	0.6262	0.6278	0.6267	0.6260	0.6272
	0.01	0.6209	0.6246	0.6163	0.6241	0.6241
	0.001	0.6312	0.6341	0.6311	0.6325	0.6297
	0.0001	0.6181	0.6191	0.6168	0.6209	0.6182
2	0.1	0.6142	0.6162	0.6126	0.6131	0.6165
	0.01	0.6180	0.6182	0.6178	0.6156	0.6169
	0.001	0.6240	0.6289	0.6260	0.6231	0.6243
	0.0001	0.6140	0.6177	0.6133	0.6159	0.6148
3	0.1	0.6152	0.6128	0.6171	0.6153	0.6145
	0.01	0.6134	0.6147	0.6133	0.6150	0.6099
	0.001	0.6094	0.6079	0.6105	0.6113	0.6063
	0.0001	0.6178	0.6185	0.6196	0.6166	0.6222

Table A.16: Sensitivity of radial basis function neural networks (RBFNN) to the number of neurons and width β. Accuracy on validation sample. Average over three runs.

Currency	β	Neurons						
		20	40	60	80	100	200	300
Bitcoin	1.0	0.5024	0.5005	0.5008	0.5004	0.5005	0.4944	0.4992
	1.5	0.4974	0.5003	0.4999	0.5009	0.5016	0.4986	0.4992
	2.0	0.4985	0.5000	0.5009	0.4978	0.4983	0.4993	0.4988
	2.5	0.5007	0.5008	0.5008	0.5028	0.5000	0.4992	0.4979
	3.0	0.4969	0.4993	0.5008	0.4998	0.4992	0.4991	0.5005
	3.5	0.5003	0.5002	0.5002	0.5001	0.5009	0.4967	0.4999
Dash	1.0	0.7072	0.7102	0.709	0.7039	0.7081	0.7097	0.7094
	1.5	0.7086	0.7102	0.7087	0.7066	0.7100	0.7118	0.7099
	2.0	0.7113	0.7071	0.7064	0.7068	0.7093	0.7092	0.7086
	2.5	0.7083	0.7093	0.7062	0.7129	0.7095	0.7077	0.7094
	3.0	0.7076	0.7093	0.7108	0.7106	0.7098	0.7099	0.7082
	3.5	0.7099	0.7070	0.7096	0.7058	0.7092	0.7086	0.7104
Ether	1.0	0.5489	0.5483	0.5444	0.5452	0.5454	0.546	0.5459
	1.5	0.5458	0.5461	0.5463	0.5449	0.5448	0.5469	0.5474
	2.0	0.5458	0.5460	0.5454	0.5454	0.5454	0.5466	0.5467
	2.5	0.5435	0.5470	0.5450	0.5471	0.5446	0.5458	0.5465
	3.0	0.5472	0.5448	0.5445	0.5460	0.5449	0.5446	0.5462
	3.5	0.5470	0.5471	0.5459	0.5457	0.5432	0.5463	0.5461
Litecoin	1.0	0.6078	0.6038	0.6065	0.6047	0.6072	0.6079	0.6064
	1.5	0.6071	0.6072	0.6076	0.6084	0.6068	0.6062	0.6084
	2.0	0.6053	0.6082	0.6066	0.6062	0.6077	0.6070	0.6065
	3.0	0.6079	0.6076	0.6069	0.6081	0.6056	0.6075	0.6066
	2.5	0.6065	0.6079	0.6061	0.607	0.6094	0.6073	0.6079
	3.5	0.6075	0.6071	0.6084	0.6058	0.6057	0.6066	0.6069
Monero	1.0	0.7433	0.746	0.7434	0.7455	0.7448	0.7442	0.7451
	1.5	0.7448	0.7444	0.7454	0.7444	0.7477	0.7446	0.7446
	2.0	0.7453	0.7455	0.7457	0.7461	0.7459	0.7460	0.7439
	2.5	0.7468	0.7463	0.7453	0.7480	0.7437	0.7401	0.7455
	3.0	0.7446	0.7450	0.7442	0.7459	0.7453	0.7447	0.7456
	3.5	0.744	0.7444	0.742	0.7451	0.7442	0.7452	0.7439
Ripple	1.0	0.6275	0.628	0.6267	0.6294	0.6288	0.6261	0.6276
	1.5	0.6300	0.6281	0.6279	0.6287	0.6287	0.6264	0.6287
	2.0	0.6264	0.6276	0.6285	0.6280	0.6276	0.6284	0.6277
	2.5	0.6267	0.6292	0.6281	0.6257	0.6263	0.6287	0.6289
	3.0	0.6268	0.6287	0.6288	0.6306	0.6277	0.6286	0.6278
	3.5	0.6283	0.6292	0.6305	0.6276	0.6290	0.6281	0.6288

Table A.17: Sensitivity of convolutional neural networks to number of lags. Accuracy on validation data.

Time period	Bitcoin	Dash	Ether	Litecoin	Monero	Ripple
15 min	0.5821	0.7120	0.5795	0.6563	0.7580	0.6312
60 min	0.5245	0.7088	0.5582	0.6109	0.7511	0.6303

Table A.18: Sensitivity of convolutional neural networks to parameter configurations for Bitcoin. Experiments based on training and validation samples.

Learn. Rate	Vert. Filters	Horiz. Filters	DNN	Xception	ResNet50
0.1	0.4955	0.4888	0.5091	0.5101	0.5005
0.01	0.4990	0.5052	0.5087	0.5111	0.5102
0.001	0.5822	0.4912	0.5011	0.5144	0.5089
0.0001	0.5123	0.5001	0.5014	0.5143	0.4924

References

Adcock, R., & Gradojevic, N. (2019). Non-fundamental, non-parametric bitcoin forecasting. *Physica A: Statistical Mechanics and its Applications*, *531*, 121727.

Armano, G., Marchesi, M., & Murru, A. (2005). A hybrid genetic-neural architecture for stock indexes forecasting. *Information Sciences*, *170*, 3–33.

Atsalakis, G. S., Atsalaki, I. G., Pasiouras, F., & Zopounidis, C. (2019). Bitcoin price forecasting with neuro-fuzzy techniques. *European Journal of Operational Research*, *276*, 770–780.

Atsalakis, G. S., & Valavanis, K. P. (2009). Surveying stock market forecasting techniques - Part II: Soft computing methods. *Expert Systems with Applications*, *36*, 5932–5941.

Baek, C., & Elbeck, M. (2015). Bitcoins as an investment or speculative vehicle? A first look. *Applied Economics Letters*, *22*, 30–34.

Bao, W., Yue, J., & Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PLOS ONE*, *12*, 1–24.

Bernardi, M., & Catania, L. (2018). The model confidence set package for r. *International Journal of Computational Economics and Econometrics*, *8*, 144–158.

Borovkova, S., & Tsiamas, I. (2018). An Ensemble of LSTM Neural Networks for High-Frequency Stock Market Classification. *Quantitative Finance, Forthcoming*, (pp. 1–27).

Campbell, J. Y., Lo, A. W. A. W.-C., & MacKinlay, A. C. (1997). *The econometrics of financial markets*. Princeton University Press.

Canziani, A., Paszke, A., & Culurciello, E. (2016). An Analysis of Deep Neural Network Models for Practical Applications. *arXiv e-prints*, (p. arXiv:1605.07678).

Chollet, F. (2016). Xception: Deep learning with depthwise separable convolutions. *CoRR*, *abs/1610.02357*.

Chong, E., Han, C., & Park, F. C. (2017). Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. *Expert Systems with Applications*, *83*, 187 – 205.

Corbet, S., Lucey, B., Urquhart, A., & Yarovaya, L. (2019). Cryptocurrencies as a financial asset: A systematic analysis. *International Review of Financial Analysis*, *62*, 182–199.

Corbet, S., Meegan, A., Larkin, C., Lucey, B., & Yarovaya, L. (2018). Exploring the dynamic relationships between cryptocurrencies and other financial assets. *Economics Letters*, *165*, 28–34.

Daníelsson, J., & Love, R. (2006). Feedback trading. *International Journal of Finance & Economics*, *11*, 35–53.

Das, S., Mokashi, K., & Culkin, R. (2018). Are markets truly efficient? experiments using deep learning algorithms for market movement prediction. *Algorithms*, *11*, 138.

Dempster, M. A. H., & Leemans, V. (2006). An automated fx trading system using adaptive reinforcement learning. *Expert Syst. Appl.*, *30*, 543–552.

Diebold, F. X., & Mariano, R. S. (1995). Comparing predictive accuracy. *Journal of Business & Economic Statistics*, *13*, 253–263.

Diler, A. (2003). Predicting direction of ISE national-100 index with back propagation trained neural network. *Journal of Istanbul Stock Exchange*, *7*,, 65–81.

Ding, X., Zhang, Y., Liu, T., & Duan, J. (2015). Deep learning for event-driven stock prediction. In *Proceedings of the 24th International Conference on Artificial Intelligence* IJCAI'15 (pp. 2327–2333). AAAI Press.

Dresp-Langley, B., Ekseth, O. K., Fesl, J., Gohshi, S., Kurz, M., & Sehring, H.-W. (2019). Occam's razor for big data? on detecting quality in large unstructured datasets. *Applied Sciences*, *9*.

Dyhrberg, A. H., Foley, S., & Svec, J. (2018). How investible is Bitcoin? Analyzing the liquidity and transaction costs of Bitcoin markets. *Economics Letters*, *171*, 140–143.

Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, *270*, 654–669.

Giacomini, R., & White, H. (2006). Tests of conditional predictive ability. *Econometrica*, *74*, 1545–1578.

Glaser, F., Zimmermann, K., Haferkorn, M., Weber, M., & Siering, M. (2014). Bitcoin - asset or currency? revealing users' hidden intentions. In *ECIS 2014 Proceedings - 22nd European Conference on Information Systems*.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

Guresen, E., Kayakutlu, G., & Daim, T. U. (2011). Using artificial neural network models in stock market index prediction. *Expert Systems with Applications*, *38*, 10389 – 10397.

Hansen, P. R., Lunde, A., & Nason, J. M. (2011). The model confidence set. *Econometrica*, *79*, 453–497.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, *abs/1512.03385*.

Hiransha, M., Gopalakrishnan, E. A., Menon, V. K., & Soman, K. P. (2018). Nse stock market prediction using deep-learning models. *Procedia Computer Science*, *132*, 1351 – 1362. International Conference on Computational Intelligence and Data Science.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*, 1735–1780.

Hoseinzade, E., & Haratizadeh, S. (2019). CNNpred: CNN-based stock market prediction using a diverse set of variables. *Expert Systems with Applications*, *129*, 273–285.

Hsu, M. W., Lessmann, S., Sung, M. C., Ma, T., & Johnson, J. E. (2016). Bridging the divide in financial market forecasting: machine learners vs. financial economists. *Expert Systems with Applications*, *61*, 215–234.

Huang, B., Huan, Y., Xu, L. D., Zheng, L., & Zou, Z. (2019). Automated trading systems statistical and machine learning methods and hardware implementation: a survey. *Enterprise Information Systems*, *13*, 132–144.

Huang, C. L., & Tsai, C. Y. (2009). A hybrid SOFM-SVR with a filter-based feature selection for stock market forecasting. *Expert Systems with Applications*, *36*, 1529–1539.

Huang, W., Nakamori, Y., & Wang, S.-Y. (2005). Forecasting stock market movement direction with support vector machine. *Computers & Operations Research*, *32*, 2513–2522.

Kara, Y., Acar Boyacioglu, M., & Baykan, O. K. (2011). Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange. *Expert Systems with Applications*, *38*, 5311–5319.

Katsiampa, P. (2017). Volatility estimation for Bitcoin: A comparison of GARCH models. *Economics Letters*, *158*, 3–6.

Kim, K.-J., & Han, I. (2000). Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index. *Expert Systems with Applications*, *19*, 125–132.

Kim, T. (2017). On the transaction cost of Bitcoin. *Finance Research Letters*, *23*, 300–305.

Kristjanpoller, W., & Minutolo, M. C. (2018). A hybrid volatility forecasting framework integrating GARCH, artificial neural network, technical analysis and principal components analysis. *Expert Systems with Applications*, *109*, 1–11.

Kwon, D.-H., Kim, J.-B., Heo, J.-S., Kim, C.-M., & Han, Y.-H. (2019). Time Series Classification of Cryptocurrency Price Trend Based on a Recurrent LSTM Neural Network. *Journal of Information Processing Systems*, *15*, 694–706.

Lahmiri, S., & Bekiros, S. (2019). Cryptocurrency forecasting with deep learning chaotic neural networks. *Chaos, Solitons & Fractals*, *118*, 35 – 40.

Li, X., & Wang, C. A. (2017). The technology and economic determinants of cryptocurrency exchange rates: The case of bitcoin. *Decision Support Systems*, *95*, 49 – 60.

Lin, M., Chen, Q., & Yan, S. (2013). Network in network. *CoRR*, *abs/1312.4400*.

Liu, D., & Zhang, L. (2010). China stock market regimes prediction with artificial neural network and markov regime switching. *WCE 2010 - World Congress on Engineering 2010*, *1*, 378–383.

Liu, W. (2019). Portfolio diversification across cryptocurrencies. *Finance Research Letters*, *29*, 200–205.

Lo, A. W., Mamaysky, H., & Wang, J. (2000). Foundations of Technical Analysis: Computational Algorithms, Statistical Inference, and Empirical Implementation. *The Journal of Finance*, *55*, 1705–1765.

Mäkinen, M., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2018). Forecasting of Jump Arrivals in Stock Prices: New Attention-based Network Architecture using Limit Order Book Data. *arXiv e-prints*, (p. arXiv:1810.10845).

Mallqui, D. C., & Fernandes, R. A. (2019). Predicting the direction, maximum, minimum and closing prices of daily bitcoin exchange rate using machine learning techniques. *Applied Soft Computing*, *75*, 596 – 606.

Miura, R., Pichl, L., & Kaizoji, T. (2019). Artificial Neural Networks for Realized Volatility Prediction in Cryptocurrency Time Series. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (pp. 165–172). Springer Verlag volume 11554 LNCS.

Moews, B., Herrmann, J. M., & Ibikunle, G. (2019). Lagged correlation-based deep learning for directional trend change prediction in financial time series. *Expert Systems with Applications*, *120*, 197–206.

Nakamoto, S. (2009). Bitcoin: a peer-to-peer electronic cash system. *Consulted*, (pp. 1–9).

Nakano, M., Takahashi, A., & Takahashi, S. (2018). Bitcoin technical trading with artificial neural network. *Physica A: Statistical Mechanics and its Applications*, *510*, 587 – 609.

Neely, C. J., Rapach, D. E., Tu, J., & Zhou, G. (2014). Forecasting the equity risk premium: The role of technical indicators. *Management Science*, *60*, 1772–1791.

Nelson, D. M., Pereira, A. C., & De Oliveira, R. A. (2017). Stock market's price movement prediction with LSTM neural networks. In *Proceedings of the International Joint Conference on Neural Networks* (pp. 1419–1426). IEEE volume 2017-May.

Patel, J., Shah, S., Thakkar, P., & Kotecha, K. (2015). Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. *Expert Systems with Applications*, *42*, 259 – 268.

Picasso, A., Merello, S., Ma, Y., Oneto, L., & Cambria, E. (2019). Technical analysis and sentiment embeddings for market trend prediction. *Expert Systems with Applications*, *135*, 60–70.

Platanakis, E., Sutcliffe, C., & Urquhart, A. (2018). Optimal vs naïve diversification in cryptocurrencies. *Economics Letters*, *171*, 93–96.

Selvin, S., Vinayakumar, R., Gopalakrishnan, E. A., Menon, V. K., & Soman, K. P. (2017). Stock price prediction using lstm, rnn and cnn-sliding window model. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)* (pp. 1643–1647).

Serrano, W. (2019). Genetic and deep learning clusters based on neural networks for management decision structures. *Neural Computing and Applications*, .

Sezer, O. B., & Ozbayoglu, A. M. (2018). Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach. *Applied Soft Computing*, *70*, 525 – 538.

Shintate, T., & Pichl, L. (2019). Trend Prediction Classification for High Frequency Bitcoin Time Series with Deep Learning. *Journal of Risk and Financial Management*, *12*, 17.

Silva de Souza, M. J., Almudhaf, F. W., Henrique, B. M., Silveira Negredo, A. B., Franco Ramos, D. G., Sobreiro, V. A., & Kimura, H. (2019). Can artificial intelligence enhance the Bitcoin bonanza. *The Journal of Finance and Data Science*, *5*, 83–98.

Singh, R., & Srivastava, S. (2017). Stock prediction using deep learning. *Multimedia Tools and Applications*, *76*, 18569–18584.

Sirignano, J., & Cont, R. (2019). Universal features of price formation in financial markets: perspectives from deep learning. *Quantitative Finance*, .

Stoye, M. (2018). Deep learning in jet reconstruction at CMS. *Journal of Physics: Conference Series*, *1085*, 042029.

Sun, J., Xiao, K., Liu, C., Zhou, W., & Xiong, H. (2019). Exploiting intra-day patterns for market shock prediction: A machine learning approach. *Expert Systems with Applications*, *127*, 272–281.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2015). Rethinking the inception architecture for computer vision. *CoRR*, *abs/1512.00567*.

Tay, F. E., & Cao, L. (2001). Application of support vector machines in financial time series forecasting. *Omega*, *29*, 309–317.

Tsantekidis, A., Passalis, N., Tefas, A., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2017). Forecasting stock prices from the limit order book using convolutional neural networks. In *2017 IEEE 19th Conference on Business Informatics (CBI)* (pp. 7–12). volume 01.

Urquhart, A. (2017). Price clustering in Bitcoin. *Economics Letters*, *159*, 145–148.

Vidal-Tomás, D., & Ibañez, A. (2018). Semi-strong efficiency of bitcoin. *Finance Research Letters*, .

Vo, A., & Yost-Bremm, C. (2018). A High-Frequency Algorithmic Trading Strategy for Cryptocurrency. *Journal of Computer Information Systems*, *4417*.

Wei, W. C. (2018). Liquidity and market efficiency in cryptocurrencies. *Economics Letters*, *168*, 21–24.

White, H. (2000). A reality check for data snooping. *Econometrica*, *68*, 1097–1126.

Wu, C. H., Lu, C. C., Ma, Y. F., & Lu, R. S. (2019). A new forecasting framework for bitcoin price with LSTM. In *IEEE International Conference on Data Mining Workshops, ICDMW* (pp. 168–175). IEEE Computer Society volume 2018-Novem.

Xu, Q., Wang, L., Jiang, C., & Zhang, X. (2019). A novel UMIDAS-SVQR model with mixed frequency investor sentiment for predicting stock market volatility. *Expert Systems with Applications*, *132*, 12–27.

Yao, J., Tan, C. L., & Poh, H.-L. (1999). Neural Networks for Technical Analysis: a Study on Klci. *International Journal of Theoretical and Applied Finance*, *02*, 221–241.

Yoshihara, A., Fujikawa, K., Seki, K., & Uehara, K. (2014). Predicting stock market trends by recurrent deep neural networks. In D.-N. Pham, & S.-B. Park (Eds.), *PRICAI 2014: Trends in Artificial Intelligence* (pp. 759–769). Cham: Springer International Publishing.

Zafeiriou, T., & Kalles, D. (2013). Short-term trend prediction of foreign exchange rates with a neural-network based ensemble of financial technical indicators. *International Journal on Artificial Intelligence Tools*, *22*.

Zhang, Z., Zohren, S., & Roberts, S. (2019). DeepLOB: Deep Convolutional Neural Networks for Limit Order Books. *IEEE Transactions on Signal Processing*, *67*, 3001–3012.