Evolving Systems manuscript No.
(will be inserted by the editor)

A dropout weight-constrained recurrent neural network model for forecasting the price of major cryptocurrencies and CCi30 index

Ioannis E. Livieris · Stavros Stavroyiannis · Emmanuel Pintelas · Theodore Kotsilieris · Panagiotis Pintelas

Abstract Cryptocurrency is widely recognized as an alternative method for paying and exchanging currency instead of using classic coins or gold; thus, it has infiltrated almost in all financial transactions worldwide. Nowadays, cryptocurrency trade constitutes one of the most popular and promising type of profitable investments. Nevertheless, this new and constantly increasing financial market is characterized by high volatility and strong fluctuations of prices over time. As a result, it is considered essential for portfolio optimization and management, the development of a forecasting model. In this work, we propose a new time-series model based on dropout weight-constrained recurrent neural networks for forecasting cryptocurrency prices and the value of CryptoCurrency index 30 (CCi30). The proposed forecasting model exploits advanced regularization techniques for reducing the fundamental problem of overfitting. More specifically, it is characterized by the imposition of box-constraints on the weights of the network for reducing the likelihood of them blowing up to unrealistic values. Additionally, the adoption of dropout technique aims to explore hard-reaching regions of the weight space and forces the weights to away from zero. The proposed forecasting model was evaluated against state-of-the-art types of neural networks and regression models for forecasting the price of the four most widely traded digital currencies and for the prediction of CCi30 index. Our conducted experimental and detailed statistical analysis demonstrate that although weight-constrained networks give significant improvements the adoption of dropout technique in weight-constrained networks provides a boost in increasing the forecasting performance.

Keywords Weight-constrained neural networks · dropout · cryptocurrency · CCi30 index · forecasting.

I.E. Livieris
Department of Mathematics, University of Patras, GR 265-00, Greece.
E-mail: livieris@upatras.gr

S. Stavroyiannis
Department of Accounting & Finance, University of the Peloponesse, GR 241-00, Greece. E-mail: s.stavroyiannis@teipel.gr

E. Pintelas
Department of Mathematics, University of Patras, GR 265-00, Greece.
E-mail: ece6835@upnet.gr

T. Kotsilieris
Department of Business Administration, University of the Peloponnese, GR 241-00, Greece. E-mail: t.kotsilieris@teipel.gr

P. Pintelas
Department of Mathematics, University of Patras, GR 265-00, Greece.
E-mail: pintelas@upatras.gr

1 Introduction

After the Global Financial Crisis of 2008-2009, along with the failure of value-at-risk type models based on the Gaussian distribution used by the Basel Committee of Banking Supervision, the confidence of the investors in the banking system and financial organizations diminished. In 2008, a team under the pseudonymous name of Satoshi Nakamoto proposed a novel and elegant solution to the double-spending problem in online payments, via a purely peer-to-peer version of electronic cash, which would allow online payments bypassing the financial institutions. Cryptocurrency is a digital exchange tool which utilizes cryptographic functions based on blockchain technology (Narayanan et al., 2016; Norman, 2017; Parker, 2018). Transparency, decentralization and immutability are some of the advantages that the blockchain framework provides on every crypto-exchange. In our days, cryptocurrency has infiltrated almost in all global financial transactions as an alternative method for paying and exchanging currency instead of using classic coins or gold.

The proposed cryptocurrency, named Bitcoin (Nakamoto, 2008), along with others that followed did not attract much attention initially; but due to the "*internet of things*" via the inter-networking of physical devices and network connectivity for collection and exchange of data, a large va-

riety of over 2000 virtual currencies has surfaced the last years with a market capitalization of US $270 billion. In fact, in a recent research, Bovaird (2017) appreciated that the cryptocurrency market has soared more than 12 times in 2017, revealing the strong and widespread growth of cryptocurrency market. Nevertheless, since the large majority of these cryptocurrencies are relatively new, there is no sufficient amount of data yet for advanced quantitative modeling or price forecasting, and they are not highly ranked towards market capitalization to be considered as market drivers. Notice that the first four cryptocurrencies, i.e. Bitcoin (BTC), Etherium (ETH), Ripple (XRP) and Litecoin (LTC) hold 67%, 8%, 4.5%, and 2% of the global cryptocurrency market capitalization, respectively, summing up to approximately 81.5%, which can be used as proxies for cryptocurrencies.

As an attempt to construct a cryptocurrency index, due to a large number of available cryptocurrencies, the CryptoCurrency index 30 (CCi30) was officially launched on 01-Jan-2017, with a starting value arbitrarily set at 100 on 01-Jan-2015. This is a rules-based index designed with the purpose to provide an objective measure of the overall growth, the daily and long-run movement of the blockchain sector, via tracking of the 30 largest cryptocurrencies regarding market capitalization. Moreover, from the diversification point of view, CCi30 can be used both as an investment tool for passive investors and as a possible industry benchmark for investment managers.

One very common way of investing in cryptocurrency, is the "*buy, hold and sell*" strategy, similar to the stock exchange and real estate investments. In this way, the investor is buying cryptocurrency with real money, he holds this currency until reaching a higher value and then it sells in order to make a profit. The investor's personal experience and the consistent watching of cryptocurrencies exchange prices can lead to some short-term profits while high amounts of profits can be achieved with accurate price predictions since the investor would buy and sell the cryptocurrency the proper time based on his predictions. However, cryptocurrency prices have by nature highly nonlinear behavior and strong fluctuations over time, which makes the performing of accurate predictions a very challenging task. For this purpose, the use of complicated mathematical formulas and methods for the development of sophisticated prediction tools can potentially assist investors to make accurate predictions and gain significant profits. Although cryptocurrency forecasting is a significant step toward portfolio optimization, existing research studies regarding cryptocurrency price forecasting are fairly limited since the market is relatively new, while most of them focus on the Bitcoin market.

Traditional time-series methods such as ARIMA (Auto-Re-gressive Integrated Moving Average) and its variations probably constitute the most famous and widely utilized methods for cryptocurrency price prediction. However, in order for these methods to be applicable, they require assumptions such as stationarity or distribution and also require data which can be broken down into noise, seasonal and trend. Therefore, they cannot depict the nonlinear and stochastic nature of cryptocurrency time-series and be effective for this task. To this end, Artificial Neural Networks (ANNs) probably constitute the most suitable methods to bypass these problems since they are highly complex non-linear systems which are specialized to solve non-linear -problems (Boufenar et al., 2018; de Campos Souza et al., 2019; Livieris et al., 2020*a*; Malekzadeh et al., 2016; Maren et al., 2014; Petridis & Kehagias, 2012; Pratama et al., 2017; Salahshour et al., 2019; Shojaie et al., 2017). Recurrent Neural Networks (RNNs) constitute a class of neural networks which are characterized by their capability for recognizing long-term dependencies. In this type of networks, the output of each layer is stored in a context layer to be looped back in along with the input from the next layer. In this sense, the RNNs gain memory of sorts; thus, they are favored over traditional feed-forward neural networks, due to the temporal nature of cryptocurrency data.

The main objective of this research is to contribute on the development of a prediction model for the cryptocurrency forecasting. Consequently, the purpose of the current research is fundamentally two-fold: Firstly, we evaluate the performance of weight-constrained recurrent neural networks (WCRNNs) for predicting cryptocurrency prices and the value f the CCi30 index. WCRNNs (Livieris, 2019*b*) are a new type of recurrent neural networks which are characterized by imposing bounds on the weights of the network. Secondly, we investigate the forecasting performance of this new type of neural networks along with the dropout technique (Srivastava et al., 2014) in order to provide a boost of the forecasting precision.

The contribution of this work is the development of an intelligent time-series model based on dropout weight-constrained recurrent neural networks. The novelty of our approach lies in exploiting the advantages of WCRNNs and dropout technique to generate good feature representations and reduce data overfitting. More analytically, the imposition of box-constraints on the connection weights reduces the likelihood that they could "blow up" to unrealistic values while the application of dropout explore hard-reaching regions of the weight space and forces the weights to away from zero. In this way, a prediction model is efficiently trained with connection weights with small values which are defined in a more uniform way. We performed a series of experiments and compared the performance of the proposed forecasting model against other state-of-the-art models for the prediction of the daily price of the four most widely traded digital currencies, i.e. BTC, ETH, XRP and LTC as well as for the prediction of the daily value of CCi30 index. Our experimental analysis demonstrates that although weight-

constrained networks give significant improvements, utilizing these new ANN-type models along with dropout technique provide a boost in reducing the generalization error. To the best of our knowledge, this is the first research devoted to the prediction of various cryptocurrencies prices and the value of CCi30 index.

Finally, it is worth mentioning that this research is focused on the development of an expert model for forecasting cryptocurrency and less on the design and implementation of profitable trading cryptocurrency system.

The remainder of the paper is organized as follows: Section 2 presents a brief survey of recent studies, regarding the application of advanced machine learning techniques in cryptocurrency forecasting. Section 3 presents the proposed dropout weight-constrained recurrent neural network model. Section 4 presents the data utilized in this study and Section 5 presents the numerical experiments. Section 7 concludes the findings of our research and provides an outline for future prospects.

2 Cryptocurrency forecasting: state of the art

During the last years, the increasing growth of information technology and digital economy has affected and transformed many sectors, including trade and finance. Nowadays, cryptocurrency trade is becoming an integral part of the global economy and constitutes a popular type of profitable investment. As a result, there is a significantly growing interest in studying the nonlinear dynamics of digital currencies, including their inherent chaoticity and fractality. The prediction of cryptocurrency prices is considered essential and constitutes a complex and challenging task in time-series forecasting. Although cryptocurrency forecasting is a significant factor for portfolio optimization, only a limited number of works have focused on this issue, especially for the Bitcoin market, which are briefly presented below.

Radityo et al. (2017) examined a number of ANN models to forecast the market value of Bitcoin. They studied four ANN models, namely genetic algorithm neural network, genetic algorithm backpropagation neural network, backpropagation neural network and neuro-evolution of augmenting topologies to predict Bitcoins' close value in the next day. All models were evaluated using the training time and the mean absolute percentage error as measurements. Their experiments showed that the backpropagation neural network demonstrated the best performance for Bitcoin prediction.

Sin & Wang (2017) studied how the features of Bitcoin such as cost per transaction, estimated transaction volume, market price and capitalization and number of transactions affect the next day's change in the movement direction of the Bitcoin price. Additionally, they proposed a new classification model called Genetic Algorithm-based Selective Neural Network Ensemble (GASNNE) which consists of five multilayer perceptrons, all with different number of nodes in the hidden layers. Their numerical experiments reported that GASNNE performed considerably well for this binary classification task reporting around $58\% - 63\%$ accuracy. Moreover, the authors claimed that the performance of GASNNE revealed that the Bitcoin's features utilized in their research contain useful information about the behavior of Bitcoin's price.

Wu et al. (2018) proposed a new framework for forecasting Bitcoin daily price using two differing Long-Short Term Memory (LSTM) neural networks models (LSTM with AR(2) and conventional LSTM). The data used in their research contained the daily price, volume and transaction historical data of 7 months (from 01-Jan-2018 to 28-Jun-2018) from which 66% were utilized for training and the rest for testing. Both LSTM models were evaluated with optimized topology and parameter settings. Their detailed experimental analysis showed the predictive power and efficiency of the LSTM with AR(2) model over plain LSTM.

Attanasio et al. (2019) investigated the efficiency of the most established classification and time-series prediction models in cryptocurrency trading by back-testing model performance. For validating each model's performance they considered a time period of eight years (from 01-Jan-2018 to 31-Dec-2018) which is characterized by heterogeneous market conditions. From their experimental analysis, the authors concluded that machine learning algorithms can offer significant insights into the cryptocurrency trading. Nevertheless, they also stated that there is no single model which can exhibit the best forecasting performance, in all situations.

Valencia et al. (2019) used social media data from Twitter along with market data for forecasting the price movement of four of the most widely traded digital currencies, namely Bitcoin, Ethereum, Ripple and Litecoin. The machine learning algorithms utilized in their research were neural networks, random forest and support vector machines. Their experimental analysis was performed using a three-phase procedure: in the first approach, each algorithm is exclusively trained with social data, in the second with market data and in the third with both social and market data. Based on their results, the authors claimed that the performance of all algorithms was increased utilizing both social and market data for the training process.

Munim et al. (2019) examined and evaluated the performance of Neural Network Auto-Regression (NNAR) and ARIMA for the prediction of next-day Bitcoin price. For this purpose, they utilized both with and without re-estimation of each forecasting model for each step. The authors utilized multiple training and testing samples to illustrate the consistency of the prediction results which revealed the superiority of ARIMA over NNAR as it was also confirmed by the Diebold-Mariano test.

Nevertheless, none of the mentioned research studies considered to develop an efficient prediction model exploiting advanced regularization techniques for addressing and reducing the fundamental problem of overfitting. Our forecasting model is characterized by the imposition of box-constraints on the weights of the network for reducing the likelihood that them could "blow up" to unrealistic values. Additionally, the adoption of dropout technique aims to explore hard-reaching regions of the weight space and forces the weights away from zero. Furthermore, unlike the previous studies, we provide extensive performance evaluation as well as a detailed statistical analysis including a nonparametric statistical and a post-hoc test to demonstrate the efficiency of our approach.

3 Proposed forecasting model

Artificial Neural Networks (ANNs) are widely characterized as one of the most powerful machine learning models, which have been successfully applied to tackle challenging real-world problems. A significant drawback of the application of ANNs is overfitting, which occurs when the network aligns too closely with the training set. This implies that although the ANN exhibits excellent performance on the training set, it performs poorly when applied to new unseen data. During the last decade, A variety of regularization strategies and approaches have been proposed in the literature to address this problem.

3.1 Dropout

Dropout (Srivastava et al., 2014) is a recently proposed regularization strategy which attempts to deal with the problem of data overfitting, without being as computationally inefficient as an ensemble of ANNs. The key idea in dropout technique is that a thinned ANN is sampled and trained at each iteration of the training process. The thinned network is created by temporary removing a random set of neurons (excluding the output neurons) along with their connection weights, with a pre-defined probability q (called dropout rate). Moreover, during testing, the output of each neuron is multiplied by the probability of being re-trained $1-q$. In this regard, an approximation to the average of the predictions of all created thinned ANNs is easily and elegantly performed.

Since at each new iteration, a randomly selected set of neurons is "dropped", the ANN is trained with fewer neurons; thus, requiring more iterations to converge. Regarding the dropout rate q, several researchers provided empirical evidence that the optimum dropout rate falls anywhere within the interval $[0,1]$ since it is heavily depended on the dataset and the network's topology (Baldi & Sadowski, 2013; Srivastava et al., 2014).

The significant advantage of the dropout strategy is that it prevents each neuron from co-adapting too much to the instances of the training set and provides a way of efficiently combining an exponential number of ANNs with different architectures. Moreover, it has the potential to reduce overfitting and provide significant improvements over other regularization strategies such as soft-weight sharing and L-regularization (Srivastava et al., 2014).

3.2 Weight-constrained neural networks

Recently, a novel approach was proposed to reduce overfitting and improve the generalization efficiency of ANNs. More specifically, Livieris (2019*a*,*b*) considered to imposition of the connection weights of the network to take certain values within pre-defined intervals; therefore, imposing box-constraints on the weights, during the training process. As a result, from a mathematical point of view, the problem of training a weight-constrained network can be formulated as a constrained optimization problem of an error function $E(w)$ which depends on the weights w of the network, namely

$$\min\{E(w)\,|\,w\in\mathscr{B}\} \tag{1}$$

with

$$\mathscr{B}=\{w\in\mathbb{R}^n : l\leq w\leq u\} \tag{2}$$

where $l=[l_1,l_2,\ldots,l_n]\in\mathbb{R}^n$ and $u=[u_1,u_2,\ldots,u_n]\in\mathbb{R}^n$ denote the vectors with the lower and upper bounds on the weights, respectively. A graphical overview of a typical WCNN with two inputs and one hidden layer is demonstrated in Figure 1.

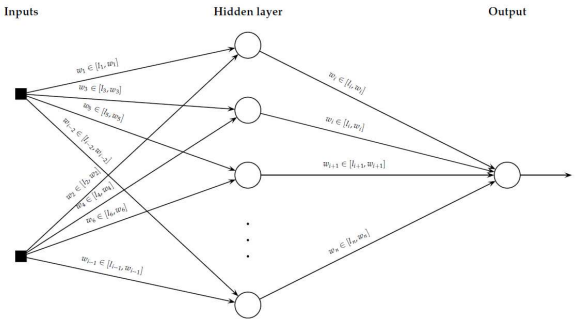

Fig. 1 A typical weight-constrained neural network with two inputs and one hidden layer

Empirical evidence in a variety of real-world benchmarks revealed that weight-constrained neural networks (WCNNs)

considerably outperform classical ANNs, in terms of generalization performance (Livieris, 2019*a*,*b*; Livieris et al., 2019*a*).

The motivation behind WCNNs is based on the fact that a trained ANN with some weights having large values is a sign that it has a very high variance and has overfitted the training data, which implies that the network is highly unstable. This instability, concerning the inputs, results in developing an unreliable prediction model since minor variations or uncertainties on the expected inputs may be greatly magnified and lead to significantly poor prediction performance on new unseen data. In contrast, a trained ANN with small weights suggests a more stable prediction model which is less likely to overfit the training data since it is less sensitive to statistical fluctuations in the inputs.

3.3 Dropout weight-constrained recurrent neural networks

The goal of this research is the development of an intelligent time-series model for forecasting cryptocurrency data. On the basis of this idea, we hybridized the previous two approaches presented by Livieris (2019*b*) and Srivastava et al. (2014) into a new forecasting model, called Dropout Weight-Constrained Recurrent Neural Network (DWCRNN). The proposed model aims on improving the efficiency of WCRNNs by adopting the dropout technique. In other words, it attempts to prevent and reduce data overfitting and provide a way of approximately combining an exponential number of different WCRNNs architectures.

Notice that WCRNNs differ from the classical feed-forward WCNN by the fact that each previous state is fed-back into the network. Therefore, the application of standard dropout to WCRNNs may tend in limiting their valuable ability of retaining memory and recognizing long-term dependencies. To address this issue, we adopt the technique proposed by Pham et al. (2014) in which the dropout is applied only to the feed-forward connections of the WCRNNs in order to improve the generalization performance. The rationale behind this technique is that by not utilizing dropout on the recurrent connections of the WCRNN, the network has a potential to benefit from dropout regularization, without sacrificing its memorization ability.

DWCRNN can be efficiently trained in a manner similar to classical WCRNNs by utilizing a classical weight-constrained training algorithm (Livieris, 2019*b*; Livieris & Pintelas, 2019). The principle difference is that at each training iteration, a thinned WCRNN is sampled by temporally dropping out feed-forward connections. For completeness, the training algorithm is presented in Algorithm 1 which is based on the improved training algorithm proposed by Livieris & Pintelas (2019). In addition, a graphical overview of the proposed training algorithm in the form of a flowchart is presented in Figure 2.

Initially, at each iteration the training algorithm temporary drops a randomly selected set of feed-forward connections, with pre-defined probability q (Step 4). Then, the algorithm calculates the Hessian approximation B_k of the error function $E(w)$ utilizing the advanced scaling factor proposed by Livieris & Pintelas (2019) (Steps 5-6). The efficiency of the algorithm heavily depends on the selection of B_k which is used to define and calculate the quadratic model $m_k(w)$ (Step 7). It worths noticing that the Hessian approximation B_k is computed utilizing the limited memory BFGS (L-BFGS) formula (Nocedal & Wright, 2006), which stores only a (relative) small number $\hat{m}$ of correction pair of vectors $\{s_k, y_k\}$. Thus, this moderate requirement of memory makes the proposed algorithm well suitable for training neural networks with especially a large number of weights.

Subsequently, the training algorithm conducts a minimization process of the approximation model $m_k(w)$, subject to the feasible domain $D = \{w \in \mathbb{R}^n \,|\, l \leq w \leq u\}$, which consists of three stages in order to compute the new vector of weights. In Stage I, a gradient projection algorithm (Morales & Nocedal, 2011) is used to compute the generalized Cauchy point w^C and calculate the set of active weights $\mathscr{A}(w^C)$ (Steps 8-9). More analytically, the generalized Cauchy point w^C is defined as a local minimum of quadratic approximation of $E(w)$, starting from the current iteration w_k, on the path defined by the projection of the steepest descent direction on the feasible domain D; while the active set $\mathscr{A}(w^C)$ is defined by the variables whose value at w^C is at lower or upper bound. In Stage II, a minimization of the approximation model $m_k(w)$ is performed on the weights characterized as non-active (i.e. $w \notin \mathscr{A}(w^C)$) utilizing a direct primal algorithm (Morales & Nocedal, 2011) (Step 10). For addressing this optimization problem, a direct primal method (Morales & Nocedal, 2011) is utilized to find the minimizer $\overline{w}_{k+1}$. In Stage III, the weights of the network are updated by performing a line search procedure along the search direction $d_k = \overline{w}_{k+1} - w_k$, which satisfies the strong Wolfe conditions (Steps 11-13). Finally, the algorithm updates the set of stored correction pairs used for the calculation of the Hessian approximation. For more information about the theoretical advantages of the scaling factor and the L-BFGS matrices as well as the minimization process, we refer the reader to Livieris (2019*b*); Livieris & Pintelas (2019); Nocedal & Wright (2006).

After the training process is complete, following the dropout technique, then the feed-forward output of each neuron is multiplied by $1-q$ (Step 17) which provides an elegant approach for approximating the average predictions of all created thinned WCRNNs.

Algorithm 1

Input: w_0 – Initial weights.
σ_1 – Hyper-parameter of strong Wolfe line search.
σ_2 – Hyper-parameter of strong Wolfe line search.
l – Vector with lower bounds on the weights.
u – Vector with upper bounds on the weights.
m – Number of stored correction vector pairs.
κ – Hyper-parameter of the scaling factor.
q – Dropout rate (in percentage).

Output: w_k – Weights of the WCRNN.

Step 1. Set $k = 0$.
Step 2. **repeat**
Step 3. Set $\hat{m} = \min\{k, m-1\}$.

Step 4. Drop a random set of feed-forward connections, with pre-defined probability q.
Step 5. Calculate the scaling parameter θ_k

$$\theta_k = \max\left\{\theta_k^{(1)}, \theta_k^{(2)}\right\}$$

where

$$\theta_k^{(1)} = \frac{s_{k-\hat{m}+1}^T y_{k-\hat{m}+1}}{y_{k-\hat{m}+1}^T y_{k-\hat{m}+1}}$$

$$\theta_k^{(2)} = \begin{cases} \dfrac{\|y_k\|^2}{s_k^T y_k}, & \text{if } \langle s_k, y_k\rangle^2 > \kappa; \\ \dfrac{y_k^T s_k}{\|s_k\|^2}, & \text{otherwise.} \end{cases}$$

$\langle\cdot,\cdot\rangle$ stands for the inner product, $s_k = w_k - w_{k-1}$ and $y_k = \nabla E(w_k) - \nabla E(w_{k-1})$.
Step 6. Calculate the Hessian approximation B_k

$$B_k = \frac{1}{\theta_k} I - \begin{bmatrix} Y_k & \frac{1}{\theta_k} S_k \end{bmatrix} \begin{bmatrix} -D_k & L_k^T \\ L_k & \frac{1}{\theta_k} S_k^T S_k \end{bmatrix}^{-1} \begin{bmatrix} Y_k & \frac{1}{\theta_k} S_k \end{bmatrix}^T,$$

where the matrices S_k, Y_k, D_k and L_k are respectively defined by

$$S_k = [s_{k-\hat{m}}, \ldots, s_{k-1}]$$
$$Y_k = [y_{k-\hat{m}}, \ldots, y_{k-1}]$$
$$D_k = diag\left[s_{k-\hat{m}}^T y_{k-\hat{m}}, \ldots, s_{k-1}^T y_{k-1}\right]$$
$$(L_k)_{ij} = \begin{cases} (s_{k-\hat{m}-1+i})^T (y_{k-\hat{m}-1+j}), & \text{if } i > j; \\ 0, & \text{otherwise.} \end{cases}$$

Step 7. Set the quadratic model $m_k(w)$ at w_k

$$m_k(w) = E(w_k) + (w - w_k)^T \nabla E(w_k) + \frac{1}{2}(w - w_k)^T B_k (w - w_k)$$

/* STAGE I */
Step 8. Calculate the generalized Cauchy point w^C.
Step 9. Define the active set $\mathscr{A}(w^C)$.

/* STAGE II */
Step 10. Minimize the quadratic model $m_k(w)$

$$\overline{w}_{k+1} = \arg\min_{w \in D_S} m_k(w)$$

where $D_S = \left\{w \in \mathbb{R} \,|\, l_i \le w_{k_i} \le u_i, \forall i \notin \mathscr{A}(w^C)\right\}$.

/* STAGE III */
Step 11. Set the search direction $d_k = \overline{w}_{k+1} - w_k$.
Step 12. Compute the learning rate η_k satisfying the strong Wolfe line search conditions

$$E_{k+1} \le E_k + c_1 \eta_k d_k^T \nabla E(w_k),$$
$$|d_k^T \nabla E(w_{k+1})| \le c_2 |d_k^T \nabla E(w_k)|,$$

using initial step size $\eta = 1$.
Step 13. Update the weights $w_{k+1} = w_k + \eta_k d_k$.
Step 14. Update the stored set of correction pairs $\{s_i, y_i\}_{i=k-\hat{m}}^{k-\hat{m}}$ satisfying $s_i^T y_i > 0$.
Step 15. Set $k = k + 1$.
Step 16. **until** (stopping criterion).
Step 17. Multiply the feed-forward connections of each neuron by $1 - q$.

4 Data

For the purpose of this research, we utilized data from 01-Jan-2017 to 30-Jun-2019, concerning the daily prices of the BTC, ETH, XRP and LTC in USD and the daily values of the CCi30 index. These cryptocurrencies were selected because at the time, they had the highest market capitalization. Moreover, the data for all cryptocurrencies were collected from https://coinmarketcap.com while for the CCi30 index from https://cci30.com.

A critical issue when dealing with the modeling of virtual currencies is the time span under consideration. Average prices across exchanges appear from Apr-2013 for BTC and LTC, Aug-2013 for XRP, Aug-2015 for ETH and Jan-2017 for the CCi30 index. Under these conditions, the starting date of our sample will be 01-Jan-2017, where the significant price increase for most of the cryptocurrencies began, to 30-Jun-2019 as the last value of the sample. Figure 3 demonstrates the daily price of the cryptocurrencies BTC, ETH, XRP and LTC and the daily value of the CCi30 index. Additionally, Table 1 presents the descriptive statistics including Mean, Median, Maximum, Minimum, Standard Deviation (Std. Dev.), Skewness and Kurtosis for each cryptocurrency and CCi30 index.

Statistic	BTC	ETH	XRP	LTC	CCi30
Mean	5743.6	318.37	0.4174	76.703	4534.2
Median	5623.5	251.76	0.3212	57.180	3718.0
Maximum	19497.0	1396.4	3.3800	358.34	20796.0
Minimum	777.76	8.17	0.0054	3.7100	276.35
Std. Dev.	3546.0	263.0	0.4025	63.597	3605.9
Skewness	0.9408	1.3628	3.2079	1.4950	1.6354
Kurtosis	4.1814	4.6885	18.313	5.3415	6.0347

Table 1 Descriptive statistics for BTC, ETH, XRP and LTC cryptocurrencies and CCi30 index

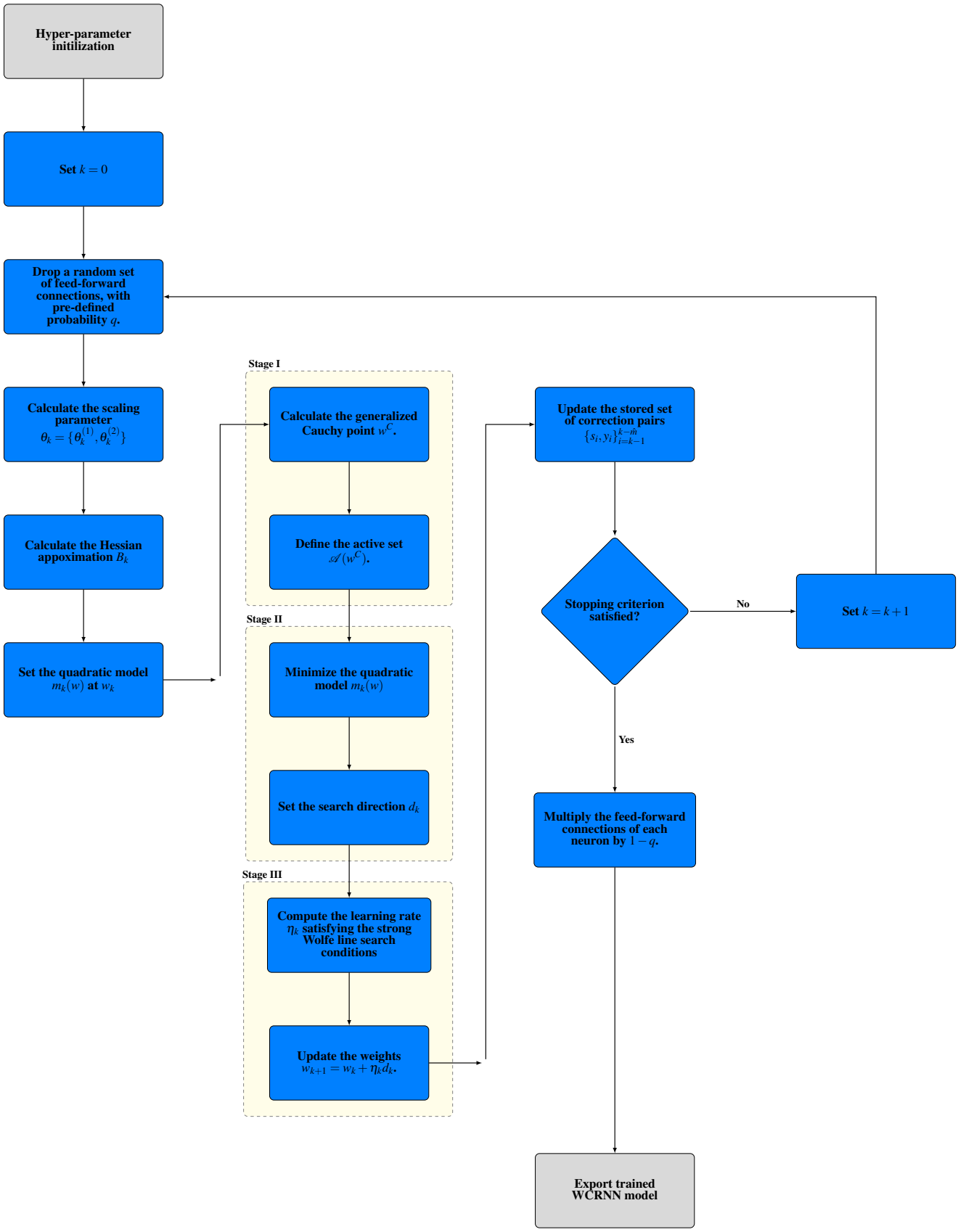

Fig. 2 Flow-chart

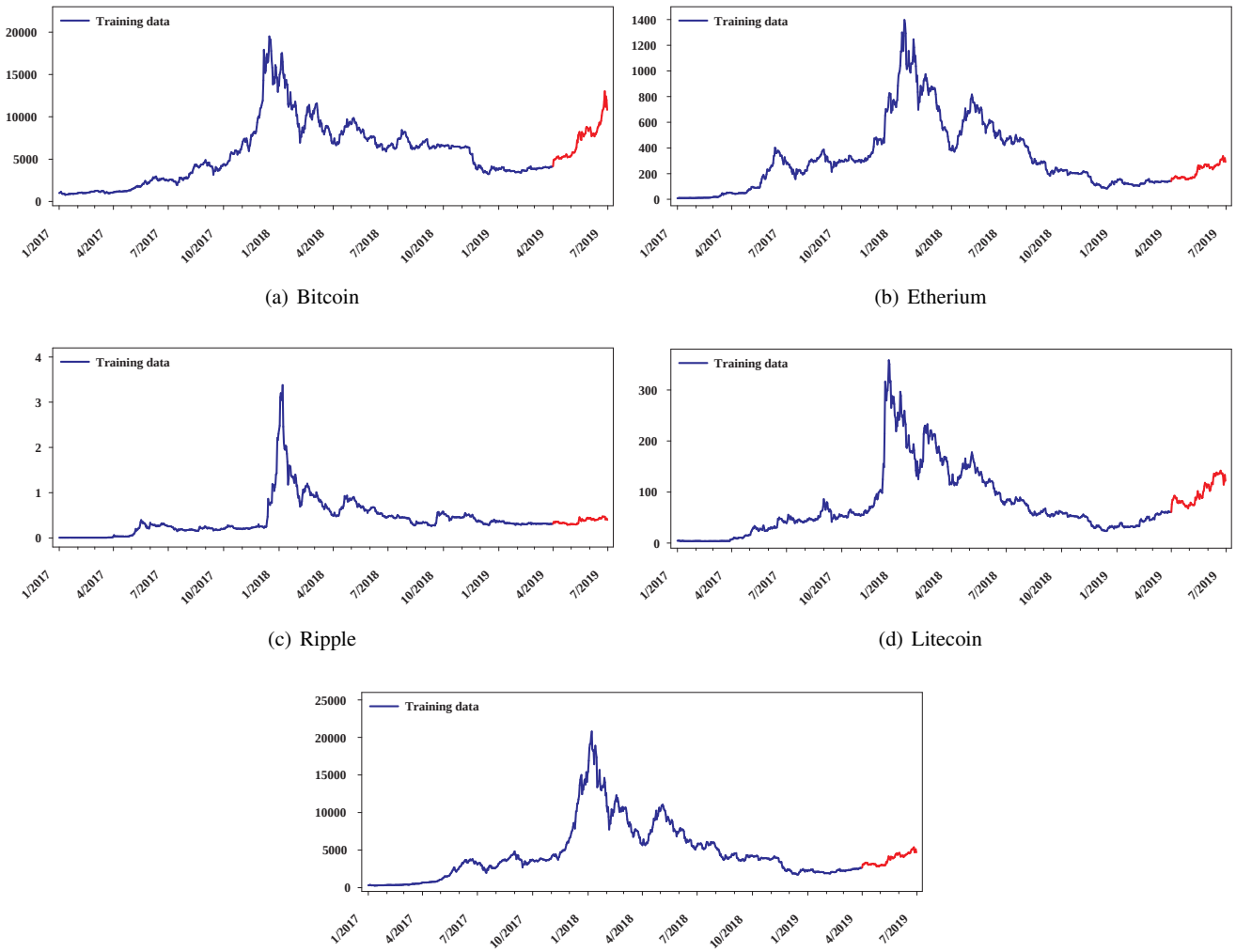

Fig. 3 Daily price of cryptocurrency BTC, ETH, XRP and LTC in USD and of CCi30 index from January 2017 to June 2019

The data were divided into training set which consists of data from 01-Jan-2017 to 31-Mar-2019 (27 months) and testing set from 01-Apr-2019 to 30-Jun-2019 (3 months). It is worth noticing that 27 months of daily values for training cover a wide range of long and short-term trends and ensures a substantial amount of data for training while the rest 3 months of daily values ensure that we evaluate the compared forecasting models on unseen "*out-of-sample*" data. Finally, in order to overcome the problems which usually arise when a series is a stochastic Brownian motion (random walk with a drift), we applied the novel methodology proposed by Livieris et al. (2020*c*) and the data were transformed via the first differences of the series to ensure stationarity.

The interpretation of Figure 2, presents that Etherium and Ripple do not have large variability as Bitcoin and Litecoin. However, since Etherium and Ripple are ranked #2 and #4 in market capitalization, respectively they are traditionally included in most research attempts in the crypto market. Moreover, Ripple is a different crypto, from the point of view that it is not mineable, it is pre-mined, and it has small variability. The CCi30 index as a weighted average of all cryptocurrencies has a reasonably high variability, which is mostly affected by Bitcoin.

5 Numerical experiments

In this section, we conducted an extensive experimental analysis to explore and examine the performance of DWCRNNs in forecasting cryptocurrency prices and also evaluate them against other state-of-the-art regression models.

The performance of each model was measured by Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) as in Aha (2013); Attanasio et al. (2019); Baldi & Sadowski

(2013); Bovaird (2017); Chai & Draxler (2014); Debelee et al. (2020), which are respectively defined by

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|p_i - a_i| \quad \text{and} \quad \text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(p_i - a_i)^2}$$

where n is the number of test instances, a_i is the actual value for i-instance and p_i is the predicted value for i-instance. It is worth mentioning that these measures were selected since they probably are the most appropriate metrics for evaluating regression models (Chai & Draxler, 2014; Livieris et al., 2019*b*).

From a forecasting aspect, the forecasting horizon F is crucial for the prediction accuracy of an intelligent model. The forecasting horizon is the number of values (days) which are taken into consideration by the model for predicting the next value. In this study, we utilized three different values for the forecasting horizon, i.e., 7, 14 and 21 days, which corresponds to 1, 2 and 3 weeks, respectively. For example, a forecasting horizon equal to 7 means that the data are being taken for 7 days and results are predicted for the 8th day.

The experimental analysis was performed following a three-phase procedure: In the first phase, we explored the sensitivity of DWCRNN to the value of the dropout parameter; in the second phase, we evaluated the performance of DWCRNN against two state-of-the-art RNN architectures used in the field of deep learning, LSTM and BiLSTM and against the novel CNN networks; in the third phase, we compared the prediction performance of the WCNNs against the other widely utilized regression algorithms. Notice that in the following tables, the best performance for each performance metric and value of parameter F is illustrated in bold.

The utilized networks consist of 1 hidden layer with 12 neurons using logistic activation function, which provided us the best forecasting performance. The hyper-parameters of Algorithm 1 were set to $\sigma_1 = 10^{-4}$, $\sigma_2 = 0.9$, $m = 7$ and $\kappa = 0.5$ (Livieris & Pintelas, 2019) and the implementation code was written in Matlab 7.6.

5.1 Sensitivity of DWCRNN to the value of dropout rate

In the sequel, we focus our interest on the experimental analysis for studying the sensitivity of DWCRNNs to the value of the dropout rate q. To this end, we tested values of q ranging from 10 to 40 in steps of 10, using two different bounds for the weights, i.e., $[-1,1]$ and $[-2,2]$ as in (Livieris, 2019*b*; Livieris & Pintelas, 2019). The abbreviations in the following tables have the following meaning

- "WCRNN" stands for Algorithm 1 with $q = 0$ (no dropout) (Livieris, 2019*b*).
- "DWCRNN (10%)" stands for Algorithm 1 with dropout rate $q = 10\%$.
- "DWCRNN (20%)" stands for Algorithm 1 with dropout rate $q = 20\%$.
- "DWCRNN (30%)" stands for Algorithm 1 with dropout rate $q = 30\%$.
- "DWCRNN (40%)" stands for Algorithm 1 with dropout rate $q = 40\%$.

For rejecting the hypothesis that all versions of Algorithm 1 performed equally well for a given level, based on their performance, we utilized the non-parametric Friedman Aligned Ranking (FAR) (Hodges & Lehmann, 1962) test. Additionally, for examining if the differences in the performance of the versions of Algorithm 1 are statistically significant, we applied the post-hoc Finner test (Finner, 1993) with significance level $\alpha = 5\%$.

Table 2 presents the detailed performance of all versions of Algorithm 1 with bounds $[-1,1]$ on the connection weights, regarding the MAE and RMSE performance metrics. Moreover, Table 3 reports the statistical analysis, performed by nonparametric multiple comparison. Firstly, it is worth noticing that the dropout technique considerably improved the performance of weight-constrained networks, as confirmed statistically by the FAR and Finner tests. Clearly, DWCRNN (10%) and DWCRNN (20%) reported the best performance, relative to all datasets and forecasting horizons. In more detail, DWCRNN (10%) and DWCRNN (20%) scored the best MAE and RMSE in 11 and 5 out of 15 cases, respectively. Moreover, the interpretation of Tables 2 and 3 suggests that for forecasting horizons 7 and 14, DWCRNN (10%) reported the lowest MAE and RMSE score; while for forecasting horizon 21, DWCRNN (20%) reported comparable and sometimes superior to that of DWCRNN (10%). This implies that for small weight-constrained networks, Algorithm 1 performs better with dropout rate $q = 10\%$; while as the size of the network increases, Algorithm 1 exhibits the better performance with dropout rate $q = 20\%$.

Table 4 and 5 report the MAE and RMSE performance and the statistical analysis of all versions of Algorithm 1 with bounds $[-2,2]$ on the weights, regarding all datasets and each value of forecasting horizon. Similar observations can be made with the previous analysis. DWCRNN (20%) exhibited the best overall forecasting performance, as it is confirmed by the statistical analysis, followed by DWCRNN (10%). More specifically, DWCRNN (20%) and DWCRNN (10%) presented the lowest MAE and RMSE in 12 and 3 out of 15 cases, respectively. Nevertheless, WCRNN considerably outperformed both DWCRNN (30%) and DWCRNN (40%) in most cases, relative to both metrics. The above analysis implies that for small values of the dropout rate q (i.e. 10% and 20%), dropout improved the performance of weight-constrained networks; in contrast, for bigger values (i.e. 30% and 40%), dropout degrades the performance of DWCRNN.

Data	F	MAE					RMSE				
		WCRNN	DWCRNN (10%)	DWCRNN (20%)	DWCRNN (30%)	DWCRNN (40%)	WCRNN	DWCRNN (10%)	DWCRNN (20%)	DWCRNN (30%)	DWCRNN (40%)
BTC	7	178.631	**147.891**	150.205	151.138	170.161	267.87	**213.305**	215.193	224.601	239.348
	14	174.133	**145.512**	157.894	161.352	175.012	247.58	**202.841**	227.946	219.991	245.151
	21	130.261	**108.951**	109.733	116.281	114.041	176.82	**151.180**	154.538	161.856	161.694
ETH	7	6.553	**5.086**	5.262	6.031	5.783	8.525	**6.873**	7.021	7.788	8.056
	14	6.142	**5.348**	5.735	5.491	6.036	8.211	**7.276**	7.601	7.421	8.064
	21	5.765	5.064	**4.894**	5.028	5.390	7.660	6.608	**6.206**	6.524	7.092
XRP	7	0.009	**0.006**	0.007	0.008	0.009	0.012	**0.010**	0.011	0.011	0.011
	14	0.007	**0.006**	**0.006**	**0.006**	0.007	0.010	0.009	**0.007**	0.009	0.010
	21	0.010	**0.004**	0.004	0.008	0.009	0.012	**0.006**	0.007	0.011	0.011
LTC	7	2.473	**1.936**	2.278	2.408	2.874	2.976	**2.490**	3.050	3.193	3.687
	14	5.179	3.456	**3.215**	3.467	3.698	6.303	4.499	**4.196**	4.468	4.741
	21	3.744	2.523	**2.308**	2.619	3.294	4.748	3.285	**2.993**	3.412	4.269
cci30	7	59.550	**50.208**	68.763	67.226	78.931	81.551	**67.421**	95.422	91.731	107.482
	14	61.053	**51.394**	54.715	80.475	44.646	79.760	**67.872**	71.441	97.551	59.212
	21	76.798	**58.211**	60.439	66.037	70.738	100.863	**73.777**	90.556	87.191	92.258

Table 2 Performance comparison based on MAE and RMSE of dropout weight-constrained recurrent neural networks with bounds $[-1,1]$ on the weights

Model	FAR	Finner post-hoc test	
		p_F-Value	Null hypothesis
DWCRNN (10%)	18.866	-	-
DWCRNN (20%)	26.2	0.356801	accepted
DWCRNN (30%)	38.766	0.016499	rejected
DWCRNN (40%)	49.2	0.000276	rejected
WCRNN	56.966	0.000007	rejected

Based on MAE metric

Model	FAR	Finner post-hoc test	
		p_F-Value	Null hypothesis
DWCRNN (10%)	8.2	-	-
DWCRNN (20%)	8.133	0.211963	accepted
DWCRNN (30%)	7.2	0.022556	rejected
DWCRNN (40%)	1.2	0.000067	rejected
WRCNN	5.266	0.000013	rejected

Based on RMSE metric

Table 3 FAR test and Finner post-hoc test comparing DWCRNN with $[-1,1]$ bounds on the weights

Conclusively, it is worth noticing that an optimum dropout rate q is really difficult (or even impossible) to be determined for all datasets. Nevertheless, we can easily observe that its selection is not only depended on the dataset but also on the pre-defined bounds on the weights. Additionally, from the interpretation of Tables 2-5, we conclude that as the size of the network increases, the gain from dropout usually increases up to a point and then it stabilizes or declines. This suggests that when the value of the dropout rate is around $10\% - 20\%$, there is a "turning point" in the forecasting performance of Algorithm 1.

5.2 Performance evaluation of DWCRNN against LSTM, BiLSTM and CNN neural networks

In the sequel, the performance of DWCRNN is compared and evaluated against that of state-of-the-art LSTM, BiLSTM and CNN, which are briefly described below:

- LSTM (Hochreiter & Schmidhuber, 1997) is an artificial neural network which is based on recurrent neural network architecture. The main difference between a classic neural network architecture and a LSTM network is that LSTM has additional feedback connections and is very suitable for making predictions on sequential data problems like time-series.
- BiLSTM (Schuster & Paliwal, 1997) is another type of RNN which aims to acquire future and past information by connecting two hidden layers of opposite directions with the same output. BiLSTM can be very useful for predicting data points values which are correlated with future and past values like in handwriting recognition where the performance for predicting a letter can be improved by gaining information from letters before and after this specific letter.
- CNN (Rawat & Wang, 2017) is a novel type of neural networks which are based on convolutional layers and are characterized by their ability of learning the internal representation of the time-series data (Livieris et al., 2020*b*). Convolutional layers apply convolution opera-

Data	F	MAE					RMSE				
		WCRNN	DWCRNN (10%)	DWCRNN (20%)	DWCRNN (30%)	DWCRNN (40%)	WCRNN	DWCRNN (10%)	DWCRNN (20%)	DWCRNN (30%)	DWCRNN (40%)
BTC	7	147.347	126.775	**122.246**	150.575	143.132	215.224	194.472	**190.762**	220.762	215.211
	14	157.451	140.214	**128.207**	146.965	181.723	225.635	198.984	**182.463**	210.475	261.587
	21	137.930	127.131	127.095	**126.466**	142.026	190.971	169.314	178.682	**165.029**	195.755
ETH	7	4.856	4.264	**3.838**	4.209	6.409	6.394	5.525	**5.115**	5.552	8.203
	14	5.242	5.856	**4.688**	5.775	5.987	7.028	7.637	**6.353**	7.439	7.975
	21	4.757	4.801	4.288	**3.870**	4.545	7.660	6.608	6.524	**5.206**	7.092
XRP	7	0.007	**0.006**	**0.006**	**0.006**	0.007	0.010	**0.009**	**0.009**	**0.009**	0.010
	14	0.006	0.005	**0.004**	0.005	0.006	0.008	0.007	**0.006**	0.007	0.008
	21	0.005	**0.004**	**0.004**	0.005	**0.004**	0.007	**0.006**	**0.006**	0.007	**0.006**
LTC	7	2.215	**2.000**	2.175	2.357	2.527	2.952	**2.659**	2.888	3.118	3.291
	14	2.561	2.401	**2.081**	3.638	3.537	3.306	3.063	**2.674**	4.520	4.530
	21	2.694	2.009	**1.773**	2.046	2.157	3.568	2.647	**2.325**	2.671	2.856
cci30	7	54.861	48.436	**45.805**	71.598	68.352	67.654	64.960	**59.636**	86.568	94.186
	14	51.712	50.162	**46.087**	61.992	73.496	67.653	64.964	**59.632**	86.562	94.184
	21	63.345	74.529	**54.097**	67.124	74.332	81.745	87.569	**70.071**	94.974	94.746

Table 4 Performance comparison based on MAE and RMSE of dropout weight-constrained recurrent neural networks with bounds $[-2,2]$ on the weights

Model	FAR	Finner post-hoc test	
		p_F-Value	Null hypothesis
DWCNN (20%)	18.133	-	-
DWCNN (10%)	28.966	0.173427	accepted
WCNN	41.6	0.004252	rejected
DWCNN (30%)	42.9	0.003712	rejected
DWCNN (40%)	58.4	0.000002	rejected

Based on MAE metric

Model	FAR	Finner post-hoc test	
		p_F-Value	Null hypothesis
DWCNN (20%)	19.066	-	-
DWCNN (10%)	26.933	0.322910	accepted
WCNN	41.866	0.005557	rejected
DWCNN (30%)	43.166	0.004912	rejected
DWCNN (40%)	58.966	0.000002	rejected

Based on RMSE metric

Table 5 FAR test and Finner post-hoc test comparing DWCRNN with $[-2,2]$ bounds on the weights

tion between the raw input data and utilize convolution kernels for producing new feature values. More specifically, a convolution kernel (filter) can be considered as a tiny window which contains coefficient values into a matrix form and "slides" all over the input matrix applying convolution operation on each subregion (patch) that this specified window "meets" across the input matrix. The convolutional layers are usually followed by a pooling layer which constitutes a subsampling technique for extracting certain values from the convolved features and produces a lower dimension matrix.

Tables 6 and 7 report the performance comparison of DWCRNN against LSTM, BiLSTM and CNN networks regarding MAE and RMSE metrics, respectively, while Table 8 reports the statistical analysis performed by nonparametric multiple comparison procedures. Notice that DWCRNN$_1$ stands for Algorithm 1 with $q = 10\%$ and bounds $[-1,1]$ on the connection weights and DWCRNN$_2$ stands for Algorithm 1 with $q = 20\%$ and bounds $[-2,2]$ on the connection weights. LSTM and BiLSTM networks consist of one layer with 50 and 2×30 units, respectively followed by a fully-connected layer of 4 neurons and an output layer of one neuron. CNN networks consist of two convolutional layers of 16 and 32 filters of size $(2,)$, respectively, followed by a max pooling layer, an dense layer of 128 neurons and an output layer of one neuron. All network utilized Rectified Linear activation function (ReLU) in all hidden layers while ADaptive Moment Estimation (ADAM) (Kingma & Ba, 2015) was utilized as training algorithm. Additionally, LSTM, BiLSTM and CNN were evaluated using dropout technique with $q = 10\%$, $q = 10\%$ and $q = 50\%$, which reported the best performance, respectively.

Both DWCRNN$_1$ and DWCRNN$_2$ reported the best overall performance, outperforming the state-of-the-art RNNs and CNNs networks, relative to all datasets and utilized forecasting horizons. Moreover, regarding the proposed model, DWCRNN$_1$ and DWCRNN$_2$ exhibited similar forecasting performance. DWCRNN$_1$ reported the best performance for BTC and ETH data while DWCRNN$_2$ reported the best for LTC and CCi30. These conclusions are also confirmed by

Data	F	LSTM	LSTM +Dropout	BiLSTM	BiLSTM +Dropout	CNN	CNN +Dropout	DWCRNN$_1$	DWCRNN$_2$
BTC	7	252.842	248.394	246.614	244.893	246.342	246.242	147.891	**122.246**
	14	250.034	249.515	240.884	244.424	246.997	246.582	145.512	**128.207**
	21	247.595	250.974	244.755	245.279	246.105	246.677	**108.951**	127.095
ETH	7	7.519	7.318	7.214	7.339	7.499	7.594	5.086	**3.838**
	14	7.380	7.329	7.416	7.329	7.728	7.496	5.348	**4.688**
	21	7.329	7.413	7.424	7.407	7.769	7.613	5.064	**4.288**
XRP	7	0.012	0.012	0.012	0.012	0.033	0.031	**0.006**	**0.006**
	14	0.012	0.012	0.012	0.012	0.052	0.059	0.006	**0.004**
	21	0.012	0.012	0.012	0.012	0.033	0.030	**0.004**	**0.004**
LTC	7	3.889	3.938	3.923	3.869	3.953	3.920	**1.936**	2.175
	14	3.949	3.874	3.850	3.874	4.014	3.986	3.456	**2.081**
	21	3.901	3.862	3.867	3.846	3.996	3.937	2.523	**1.773**
cci30	7	111.864	112.007	113.157	109.246	112.773	112.763	50.208	**45.805**
	14	115.111	111.543	117.348	112.924	113.000	112.684	51.394	**46.087**
	21	113.425	111.514	113.963	112.246	113.098	112.709	58.211	**54.097**

Table 6 Performance evaluation of DWCRNNs with state-of-the-art LSTM, BiLSTM and CNN based on MAE

Data	F	LSTM	LSTM +Dropout	BiLSTM	BiLSTM +Dropout	CNN	CNN +Dropout	DWCRNN$_1$	DWCRNN$_2$
BTC	7	404.186	398.866	387.036	391.455	393.866	392.983	213.305	**190.762**
	14	403.286	401.771	390.086	391.642	393.280	392.740	202.841	**182.463**
	21	391.638	392.752	391.277	390.186	392.874	392.980	**151.180**	178.682
ETH	7	10.919	10.615	10.369	10.610	10.621	10.745	6.873	**5.115**
	14	10.691	10.623	10.683	10.640	10.912	10.705	7.276	**6.353**
	21	10.625	10.696	10.727	10.703	10.881	10.814	6.608	**6.524**
XRP	7	0.018	0.018	0.018	0.018	0.360	0.035	0.010	**0.009**
	14	0.018	0.018	0.018	0.018	0.057	0.062	0.009	**0.006**
	21	0.018	0.018	0.018	0.018	0.037	0.034	**0.006**	**0.006**
LTC	7	5.389	5.457	5.448	5.330	5.418	5.288	**2.490**	2.888
	14	5.494	5.354	5.274	5.420	5.458	5.403	4.499	**2.674**
	21	5.405	5.340	5.355	5.313	5.407	5.307	3.285	**2.325**
cci30	7	163.877	160.329	163.726	158.859	161.669	161.941	67.42	**59.636**
	14	163.605	161.064	167.441	163.564	161.966	161.829	67.87	**59.632**
	21	163.019	162.166	164.605	162.361	161.889	161.778	73.77	**70.071**

Table 7 Performance evaluation of DWCRNNs with state-of-the-art LSTM, BiLSTM and CNN based on RMSE

the statistical analysis. More specifically, the interpretation of Table 8 presents that DWCRNN$_2$ exhibited the highest probability-based ranking, reporting slightly better performance compared to DWCRNN$_1$.

5.3 Performance evaluation of DWCRNN against state-of-art regression algorithms

Next, we compared the forecasting performance of algorithm DWCRNNs against the state-of-the-art regression algorithms: Support Vector Regression (SVR), k-Nearest Neighbor Regression (kNN), Decision Tree Regressor (DTR) and Linear Regression (LR) which are briefly described below:

- SVR (Deng et al., 2012) is a machine learning regression algorithm which is used for forecasting continues output values, in contrast to SVM which is a classification algorithm for predicting discrete values. The main objective of SVR is to fit the error within a specified threshold instead of other classical regression algorithms like linear regression which try to minimize the error rate.

Model	FAR	Finner post-hoc test	
		p_F-Value	Null hypothesis
$DWCRNN_2$	14.466		-
$DWCRNN_1$	16.533	0.870749	accepted
BiLSTM+Dropout	71.966	0.000007	rejected
LSTM+Dropout	73.966	0.000005	rejected
BiLSTM	74.100	0.000005	rejected
LSTM	76.166	0.000003	rejected
CNN+Dropout	77.533	0.000002	rejected
CNN	79.266	0.000002	rejected

Based on MAE metric

Model	FAR	Finner post-hoc test	
		p_F-Value	Null hypothesis
$DWCRNN_2$	15.233		-
$DWCRNN_1$	17.100	0.883162	accepted
BiLSTM+Dropout	72.767	0.000007	rejected
BiLSTM	74.300	0.000006	rejected
LSTM+Dropout	74.366	0.000006	rejected
CNN+Dropout	75.133	0.000006	rejected
LSTM	76.833	0.000005	rejected
CNN	78.266	0.000005	rejected

Based on RMSE metric

Table 8 FAR test and Finner post-hoc test comparing DWCRNN with state-of-the-art RNNs

– *k*NN (Aha, 2013) is another machine learning algorithm which makes use of various distance mathematic formulas to compute feature similarity between each new instance point and a predefined number of other instance points. For classification tasks, the value of each new instance point is defined by the instance point with the greatest feature similarity (nearest neighbor) to it while for regression tasks its value is defined by the average value of its nearest neighbors.

– DTR (Loh, 2014) is a decision tree regression technique which constructs a model tree based on splitting criterions. The last nodes (leafs) of the tree have linear regression algorithms to predict the output continuous values, in contrast to the classification decision tree whose leafs have the output predicted discrete variable.

– LR (Seber & Lee, 2012) constitutes the traditional and most widely utilized type of predictive analysis. The main idea behind LR is to determine the relationship between a dependent variable and one or more explanatory (independent) variables following the linear mathematical model.

Table 9 reports the configuration parameters of all state-of-the-art regression algorithms under consideration.

Algorithm	Parameters
SVR	$C = 1.0$, Tolerance parameter $= 0.001$, Kernel type : Radial Basis Function.
*k*NN	Number of neighbors $= 10$, Euclidean distance.
DTR	Spliting critirion: MSE, Min. number of samples $= 10$.
LR	No parameters specified.

Table 9 Parameter specification for all state-of-the-art regression algorithms used in the experimentation

Table 10 presented of the proposed forecasting model DWCRNN against the state-of-the-art regression algorithms based on MAE and RMSE metrics. Clearly, $DWCRNN_1$ and $DWCRNN_2$ reported the best overall performance, reporting the lowest MAE and RMSE, regarding all data and utilized forecasting horizons.

Table 11 demonstrates the statistical analysis regarding the forecasting performance of DWCRNN and state-of-the-art regression models. Both $DWCRNN_1$ and $DWCRNN_2$ exhibited the highest probability-based ranking, outperforming all other regression models.

6 Discussion

In this work, we proposed a new time-series model based on dropout weight-constrained recurrent neural networks for forecasting major cryptocurrency prices and predicting the value of CCi30 index. Our conducted experimental analysis demonstrated that the proposed model reported considerably better performance compared to state-of-the-art types of ANNs and regression models and the adoption of dropout technique in weight-constrained networks provides a boost in increasing the forecasting performance.

It is worth mentioning that the utilized datasets in this research are characterized by the presence of large amount of noise since cryptocurrency prices follow almost a random walk process (Livieris et al., 2020*c*). In noisy datasets, a machine learning model which suffers by the overfitting problem, such as neural networks, will capture and learn noise instead of useful and reliable patterns and thus will lead to an obvious performance degradation. Based on our experimental results, the incorporation of the dropout technique into the weight-constrained neural networks has led to a noticeable performance increase (Tables 2 and 4). Furthermore, the adoption of the dropout weight-constrained methodology into neural networks, managed to drastically increase the performance and outperform other types of neural networks such as LSTM and CNN. This leads to the conclusion

Data	F	MAE						RMSE					
		SVR	kNN	DTR	LR	DWCRNN$_1$	DWCRNN$_2$	SVR	kNN	DTR	LR	DWCRNN$_1$	DWCRNN$_2$
BTC	7	244.166	255.326	321.299	250.681	147.891	**122.246**	391.170	412.411	471.951	406.459	213.305	**190.762**
	14	244.093	270.470	380.458	248.619	145.512	**128.207**	391.129	389.805	547.294	406.332	202.841	**182.463**
	21	244.108	259.163	306.993	251.084	**108.951**	127.095	391.133	364.113	436.137	410.734	**151.180**	178.682
ETH	7	7.472	8.555	12.695	7.656	5.086	**3.838**	10.721	12.108	17.758	11.024	6.873	**5.115**
	14	7.475	8.047	11.215	8.243	5.348	**4.688**	10.724	11.116	14.658	11.416	7.276	**6.353**
	21	7.475	7.797	11.641	8.682	5.064	**4.288**	10.721	11.088	15.140	12.084	6.608	**6.524**
XRP	7	0.020	0.013	0.016	0.013	**0.006**	**0.006**	0.025	0.019	0.023	0.018	0.010	**0.009**
	14	0.019	0.014	0.017	0.014	0.006	**0.004**	0.023	0.020	0.024	0.020	0.009	**0.006**
	21	0.025	0.013	0.015	0.015	**0.004**	**0.004**	0.030	0.019	0.021	0.021	**0.006**	**0.006**
LTC	7	3.867	3.914	6.024	4.055	**1.936**	2.175	5.354	5.411	7.954	5.483	**2.490**	2.888
	14	3.843	4.042	5.774	4.185	3.456	**2.081**	5.303	5.440	7.790	5.611	4.499	**2.674**
	21	3.836	3.963	5.282	4.159	2.523	**1.773**	5.291	5.309	7.528	5.559	3.285	**2.325**
cci30	7	111.835	122.920	146.032	120.919	50.208	**45.805**	161.022	168.753	192.402	170.457	67.42	**59.636**
	14	111.795	119.167	171.612	124.309	51.394	**46.087**	160.975	170.874	230.921	173.098	67.87	**59.632**
	21	111.773	120.102	164.451	126.015	58.211	**54.097**	160.953	166.720	221.347	178.665	73.77	**70.071**

Table 10 Performance evaluation of DWCRNNs with state-of-the-art regression algorithms based on MAE and RMSE

Model	FAR	Finner post-hoc test	
		p_F-Value	Null hypothesis
DWCNN$_2$	16.8	-	-
DWCNN$_1$	18.6	0.850335	accepted
SVR	51.333	0.000368	rejected
LR	57.933	0.000038	rejected
kNN	58.066	0.000038	rejected
DTR	70.266	0.0	rejected

Based on MAE metric

Model	FAR	Finner post-hoc test	
		p_F-Value	Null hypothesis
DWCNN$_2$	14.466	-	-
DWCNN$_1$	16.533	0.828485	accepted
SVR	56.6	0.000013	rejected
LR	58.066	0.000008	rejected
kNN	59.666	0.000005	rejected
DTR	67.666	0.0	rejected

Based on RMSE metric

Table 11 FAR test and Finner post-hoc test comparing DWCRNN with state-of-the-art regression algorithms

that the proposed model managed to capture and learn reliable patterns, filtering out the noisy ones and thus managed to constrain and reduce the overfitting effect.

The limitation of this work is that it is still not clear whether the use of both weight-constraints and dropout at the same time acts synergistically or makes things more complicated for no net gain. While the imposition of bounds on the weights of the network is implemented with clearly predefined intervals, dropout cannot be coherently expressed in a similar way since it requires a random process of temporary dropping off some units and therefore cannot be analyzed, other than experimentally. Nevertheless, both regularization methods attempt to avoid the network's over-reliance on spurious correlations, which are one of the consequences of over-training, based on its own philosophy and technique. Clearly, more research is needed to determine whether and when they can be efficiently "*applied together*" or rather end up "*fighting each other*". Based on our preliminary numerical experiments, it seems both weight-constraints and dropout can be efficiently combined.

Nevertheless, although we provide thoroughly experimental results revealing that our model is superior comparing to other state-of-the-art forecasting models, there are several reasons why trend prediction efforts, such as the ones described in this research, might not necessarily translate into profits. In other words, even though the presented numerical experiments are promising, we have no evidence if our model can actually assist cryptocurrency investors for making proper investment decisions based on our model predictions in order to achieve profitable investment returns, since cryptocurrency prices are highly affected by time evolution and external changes and therefore an efficient prediction model may be temporally accurate but not in long-term future. More specifically, the unpredictable cryptocurrency market changes, as well as the possible entrance of high capitalization member such as Facebook and European Central Bank, could significantly change the behavior and the variability of cryptocurrencies.

This research is focused on the performance of the proposed model in price forecasting and less on the design and implementation of profitable trading cryptocurrency system.

It is worth noticing that the development of such system would require the control of several aspects such as time management, transaction costs, liquidity issues and so on in addition to the implementation of a decision support model. Therefore, a possible improvement of our prediction framework could be a dynamic modeling approach where our model will be dynamically re-training on most recent cryptocurrency data while old and outdated data values will be discarded. Finally, we may also incorporate trading simulations in order to identify potential profitable investment returns.

7 Conclusions

At present, cryptocurrency market constitutes one of the most popular and promising type of profitable investments. The contribution of this work was to develop an intelligent forecasting model for forecasting cryptocurrency-related data. To this end, we proposed a new time-series model based on dropout weight-constrained recurrent neural networks. To the best of our knowledge, this is the first research devoted to the prediction of cryptocurrencies prices and the value of CCi30 index. The proposed forecasting model was evaluated against state-of-the-art types of ANNs and regression models for predicting the price of four of the most widely traded digital currencies and for the prediction of CCi30 index. Our performed experimental analysis illustrated that although weight-constrained networks give significant improvements, utilizing them along with dropout provide a boost in increasing the forecasting performance. A possible reason is that the "noise" which is generated by dropout allows the minimization process to explore regions of the weight space which would have been hard to reach.

In our future work, we intend to explore the efficiency of WCNNs with variants of the dropout technique (Gal & Ghahramani, 2016; Moon et al., 2015) and enforce our proposed framework with more advanced and complex techniques such as convolutional layers (Debelee et al., 2020; Rawat & Wang, 2017). Since our experimental results are quite encouraging, a possible next step could be the application of our proposed model for forecasting cryptocurrency prices using higher-frequency data (hourly, 15-min and 5-min) and other factors such as opening price, closing price, lowest price and highest price, volume and transaction along with the daily price. Finally, another interesting idea is to develop an adaptive strategy to auto-adjust the bounds on the weights based on the dropout rate.

References

Aha, D. W. (2013), *Lazy learning*, Springer Science & Business Media.

Attanasio, G., Cagliero, L., Garza, P. & Baralis, E. (2019), Quantitative cryptocurrency trading: exploring the use of machine learning techniques, *in* 'Proceedings of the 5th Workshop on Data Science for Macro-modeling with Financial and Economic Datasets', ACM, p. 1.

Baldi, P. & Sadowski, P. J. (2013), Understanding dropout, *in* 'Advances in neural information processing systems', pp. 2814–2822.

Boufenar, C., Batouche, M. & Schoenauer, M. (2018), 'An artificial immune system for offline isolated handwritten arabic character recognition', *Evolving Systems* **9**(1), 25–41.

Bovaird, C. (2017), 'Why the crypto market has appreciated more than 1,200% this year', *Forbes Magazine* .

Chai, T. & Draxler, R. R. (2014), 'Root mean square error (RMSE) or mean absolute error (MAE)?–Arguments against avoiding RMSE in the literature', *Geoscientific model development* **7**(3), 1247–1250.

de Campos Souza, P. V., Nunes, C. F. G., Guimares, A. J., Rezende, T. S., Araujo, V. S. & Arajuo, V. J. S. (2019), 'Self-organized direction aware for regularized fuzzy neural networks', *Evolving Systems* pp. 1–15.

de Campos Souza, P. V., Soares, E. A., Guimarães, A. J., Araujo, V. S., Araujo, V. J. S. & Rezende, T. S. (2020), 'Autonomous data density pruning fuzzy neural network for optical interconnection network', *Evolving Systems* pp. 1–13.

Debelee, T. G., Schwenker, F., Ibenthal, A. & Yohannes, D. (2020), 'Survey of deep learning in breast cancer image analysis', *Evolving Systems* **11**(1), 143–163.

Deng, N., Tian, Y. & Zhang, C. (2012), *Support vector machines: optimization based theory, algorithms, and extensions*, Chapman and Hall/CRC.

Finner, H. (1993), 'On a monotonicity problem in step-down multiple test procedures', *Journal of the American Statistical Association* **88**(423), 920–923.

Gal, Y. & Ghahramani, Z. (2016), A theoretically grounded application of dropout in recurrent neural networks, *in* 'Advances in neural information processing systems', pp. 1019–1027.

Hochreiter, S. & Schmidhuber, J. (1997), 'Long short-term memory', *Neural computation* **9**(8), 1735–1780.

Hodges, J. L. & Lehmann, E. L. (1962), 'Rank methods for combination of independent experiments in analysis of variance', *The Annals of Mathematical Statistics* **33**(2), 482–497.

Kingma, D. P. & Ba, J. (2015), Adam: A method for stochastic optimization, *in* 'The international conference on learning representations'.

Livieris, I. (2019*a*), Improving the classification efficiency of an ANN utilizing a new training methodology, *in* 'Informatics', Vol. 6, Multidisciplinary Digital Publishing Institute, p. 1.

Livieris, I. E. (2019*b*), 'Forecasting economy-related data utilizing weight-constrained recurrent neural networks', *Algorithms* **12**(4), 85.

Livieris, I. E., Iliadis, L. & Pintelas, P. (2020*a*), 'On ensemble techniques of weight-constrained neural networks', *Evolving Systems* pp. 1–13.

Livieris, I. E., Kotsilieris, T., Stavroyiannis, S. & Pintelas, P. (2019*a*), 'Forecasting stock price index movement using a constrained deep neural network training algorithm', *Intelligent Decision Technologies* .

Livieris, I. E., Pintelas, E., Kotsilieris, T., Stavroyiannis, S. & Pintelas, P. (2019*b*), 'Weight-constrained neural networks in forecasting tourist volumes: A case study', *Electronics* **8**(9), 1005.

Livieris, I. E., Pintelas, E. & Pintelas, P. (2020*b*), 'A cnn–lstm model for gold price time-series forecasting', *Neural Computing and Applications* pp. 1–10.

Livieris, I. E. & Pintelas, P. (2019), 'An improved weight-constrained neural network training algorithm', *Neural Computing and Applications* pp. 1–9.

Livieris, I. E., Stavroyiannis, S., Pintelas, E. & Pintelas, P. (202*c*0), 'A novel validation framework to enhance deep learning models in time-series forecasting', *Neural Computing and Applications* pp. 1–19.

Loh, W.-Y. (2014), 'Classification and regression tree methods', *Wiley StatsRef: Statistics Reference Online* .

Malekzadeh, M., Sadati, J. & Alizadeh, M. (2016), 'Adaptive PID controller design for wing rock suppression using self-recurrent wavelet neural network identifier', *Evolving Systems* **7**(4), 267–275.

Maren, A. J., Harston, C. T. & Pap, R. M. (2014), *Handbook of neural computing applications*, Academic Press.

Moon, T., Choi, H., Lee, H. & Song, I. (2015), Rnndrop: A novel dropout for RNNs in ASR, *in* '2015 IEEE Workshop on Automatic Speech Recognition and Understanding', IEEE, pp. 65–70.

Morales, J. L. & Nocedal, J. (2011), 'Remark on "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization".', *ACM Transaction of Mathematical Software* **38**(1), 1–7.

Munim, Z. H., Shakil, M. H. & Alon, I. (2019), 'Next-day Bitcoin price forecast', *Journal of Risk and Financial Management* **12**(2), 103.

Nakamoto, S. (2008), 'Bitcoin: A peer-to-peer electronic cash system', *Consulted* .

Narayanan, A., Bonneau, J., Felten, E., Miller, A. & Goldfeder, S. (2016), *Bitcoin and cryptocurrency technologies: a comprehensive introduction*, Princeton University Press.

Nocedal, J. & Wright, S. (2006), *Numerical optimization*, Springer Science & Business Media.

Norman, A. T. (2017), *Cryptocurrency Investing Bible: The Ultimate Guide About Blockchain, Mining, Trading, ICO, Ethereum Platform, Exchanges, Top Cryptocurrencies for Investing and Perfect Strategies to Make Money*, CreateSpace Independent Publishing Platform.

Parker, J. F. (2018), *Blockchain technology simplified: the complete guide to blockchain management, mining, trading and investing cryptocurrency*, CreateSpace Independent Publishing Platform.

Petridis, V. & Kehagias, A. (2012), *Predictive modular neural networks: applications to time series*, Vol. 466, Springer Science & Business Media.

Pham, V., Bluche, T., Kermorvant, C. & Louradour, J. (2014), Dropout improves recurrent neural networks for handwriting recognition, *in* '14th International Conference on Frontiers in Handwriting Recognition', IEEE, pp. 285–290.

Pratama, M., Angelov, P. P., Lu, J., Lughofer, E., Seera, M. & Lim, C. P. (2017), A randomized neural network for data streams, *in* '2017 international joint conference on neural networks (IJCNN)', IEEE, pp. 3423–3430.

Radityo, A., Munajat, Q. & Budi, I. (2017), Prediction of Bitcoin exchange rate to american dollar using artificial neural network methods, *in* '2017 International Conference on Advanced Computer Science and Information Systems (ICACSIS)', IEEE, pp. 433–438.

Rawat, W. & Wang, Z. (2017), 'Deep convolutional neural networks for image classification: A comprehensive review', *Neural computation* **29**(9), 2352–2449.

Salahshour, E., Malekzadeh, M., Gholipour, R. & Khorashadizadeh, S. (2019), 'Designing multi-layer quantum neural network controller for chaos control of rod-type plasma torch system using improved particle swarm optimization', *Evolving Systems* **10**(3), 317–331.

Schuster, M. & Paliwal, K. K. (1997), 'Bidirectional recurrent neural networks', *IEEE Transactions on Signal Processing* **45**(11), 2673–2681.

Seber, G. A. & Lee, A. J. (2012), *Linear regression analysis*, Vol. 329, John Wiley & Sons.

Shojaie, A. A., Zand, A. D. & Vafaie, S. (2017), 'Calculating production by using short term demand forecasting models: a case study of fuel supply system', *Evolving Systems* **8**(4), 271–285.

Sin, E. & Wang, L. (2017), Bitcoin price prediction using ensembles of neural networks, *in* '2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)', IEEE, pp. 666–671.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014), 'Dropout: a simple way to prevent neural networks from overfitting', *The journal of machine learning research* **15**(1), 1929–1958.

Valencia, F., Gómez-Espinosa, A. & Valdés-Aguirre, B. (2019), 'Price movement prediction of cryptocurrencies using sentiment analysis and machine learning', *Entropy*

21(6), 589.

Wu, C.-H., Lu, C.-C., Ma, Y.-F. & Lu, R.-S. (2018), A new forecasting framework for Bitcoin price with LSTM, *in* '2018 IEEE International Conference on Data Mining Workshops (ICDMW)', IEEE, pp. 168–175.