IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 16 May 2023, accepted 2 June 2023, date of publication 7 June 2023, date of current version 14 June 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3283571

RESEARCH ARTICLE

An Elitist Artificial Electric Field Algorithm Based Random Vector Functional Link Network for Cryptocurrency Prices Forecasting

SARAT CHANDRA NAYAK[1], SUBHRANGINEE DAS[2], SATCHIDANANDA DEHURI[3], (Senior Member, IEEE), AND SUNG-BAE CHO[1], (Senior Member, IEEE)

[1]Department of Computer Science, Yonsei University, Seoul 03722, South Korea
[2]Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Hyderabad 500075, India
[3]Department of Computer Science, Fakir Mohan University, Balasore 756019, India

Corresponding author: Sung-Bae Cho (sbcho@yonsei.ac.kr)

This work was supported by the Yonsei Fellow Program funded by Lee Youn Jae, Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government, Ministry of Science and ICT (MSIT) (No. 2020-0-01361, Artificial Intelligence Graduate School Program (Yonsei University); No. 2022-0-00113, Developing a Sustainable Collaborative Multi-modal Lifelong Learning Framework). Prof. Satchidananda Dehuri acknowledge the support of Teachers Associateship for Research Excellence (TARE) Fellowship (No. TAR/2021/00006) of the Science and Engineering Research Board (SERB), Government of India.

ABSTRACT Cryptocurrencies have carved out a significant presence in financial transactions during the past few years. Cryptocurrency market performs similarly to other financial markets with considerable nonlinearity and volatility and its prediction is a growing research area. It is challenging to capture the inherent uncertainties connected with cryptocurrency using the currently used conventional methodologies. The popularity of random vector functional link networks (RVFLN) is attributed to its simple structural layout, quick rate of learning, and enhanced generalization ability. It computes the output layer weights using non-iterative techniques like least square methods or iterative techniques like gradient methods, and assigns hidden neuron parameters at random. Random initialization of non-optimal hidden neuron settings, however, degrades the performance. Population-based metaheuristics are a superior option to random initialization for determining the ideal parameters and avoiding the problem of local optima stagnation. In the current article, an elitist artificial electric field algorithm (eAEFA) for training RVFLN is proposed. Here, eAEFA is utilized to create an ideal RVFLN by determining the weights and biases of the hidden layer connections. The elitism method is used by AEFA to maximize its strength. Here, the most suitable entities are directly inserted to create the population of the following generation. By predicting the closing values of six widely used cryptocurrencies, including Bitcoin, Litecoin, Ethereum, ZEC, XLM, and Ripple, one may determine how well the eAEFA+RVFLN model is performing. For comparison study, models including ARIMA, multi-layer perceptron (MLP), basic RVFLN, support vector regression (SVR), LSTM, GA trained RVFLN, and AEFA trained RVFLN are also constructed concurrently. In terms of performance and statistical significance testing, the suggested eAEFA+RVFLN findings outperform the comparator models. On an average, it achieves a MAPE (mean absolute percentage of error) value of 0.0573, R^2 (coefficient of determination) of 0.9589, POCID (prediction of change in direction) of 0.9676, RMSE (root mean squared error) of 0.0685, MAE (mean absolute error) of 0.0727 and an average rank of 1.346; as a result, it is possible to recommend it as a useful financial forecasting tool.

INDEX TERMS Cryptocurrency, bitcoin, random vector functional link network, financial time series forecasting, artificial neural network, AEFA.

The associate editor coordinating the review of this manuscript and approving it for publication was Chun-Wei Tsai.

I. INTRODUCTION

Avirtual currency produced via encryption techniques is called cryptocurrency. Due to its sharply increasing price in a short period of time, this digital money become very popular. Despite the fact that cryptocurrency is unregulated, many nations accept a variety of types of it for usage in commerce. These currencies were developed without the use of a central authority on a distributed, decentralized peer-to-peer network, which ensures pseudo-anonymity and double spending attack prevention to avoid disputes [1]. Among the available cryptocurrencies, Bitcoin is most well-known [2]. They have a large user base and expanding market share and their worth fluctuates based on the market. Therefore, investors and experts are concerned about being able to accurately estimate their market worth. The crypto market's various trading opportunities and advantages are reviewed in [3]. An interesting explanation about cryptocurrency systems is available in the literature [4]. The literature takes into account a variety of market-controlling characteristics, including high unpredictability, decentralized governance, modest capitalization, and data accessibility [5]. Even though this market has inherent volatility like other stock markets, investors' conviction may be seen in it [6], [7]. The cryptocurrency market's viability in comparison to other active financial markets is well-documented in [8] and [9]. It is clear that cryptocurrency market performs with considerable nonlinearity and volatility similar to other financial markets.

It is challenging to capture the inherent uncertainties connected with cryptocurrency using the currently used conventional methodologies. The literature suggests a number of computationally sophisticated models that capture the underlying non-linearity of cryptocurrencies. However, there is still a need to investigate a computationally efficient model with acceptable accuracy. Random forest method is found accurate at predicting high frequency Bitcoin data on hourly and daily forecast horizons [10] when compared with feedforward ANN, SVM, random walk and ARIMA model. However, the model performance is not evaluated on other cryptocurrencies available in the market. Historical market prices of four cryptocurrencies are modelled using an ensemble of ANN models such as convolutional neural network and bidirectional LSTM and the proposed ensemble model yielded better predictions over individual models [11]. A reconstructed dynamic boundary ANN model is developed by authors in [12] and applied on prediction of Bitcoin. Deep ANN models are developed using historical prices, social media and trading features of Ethereum and Bitcoin and have shown significant prediction accuracy [13]. A comparison framework of statistical, machine learning, deep learning, and ensemble-based models is proposed in [14] where, the deep learning-based models are found as the best predictor. The next section shading lights on several computational intelligent methods applied so far on cryptocurrency predictions, their advantages and limitations. Probably ANNs are the efficient mechanisms which deal the nonlinearity exhibited by the financial data accurately. As can be shown, the network architecture and learning technique used to determine the ideal ANN parameters have a significant impact on its accuracy. Though ANN models have several drawbacks like; suffering from slow rate of convergence, more computational cost; but they are still considered as a better method for crypto prediction. Training of an ANN is challenging because it needs thousands of iterations to converge. Due to the local minima issue, it might even fail to converge in some circumstances. Besides, all these drawbacks, ANN requires many parameters selection before execution.

Higher order ANNs are good enough to avoid the drawbacks associated with ANN at the same time capable of keeping all the advantages that ANNs have. RVFLN belongs to class of higher order ANN and handles the associated nonlinearity in the dataset through auto enhancement of input nodes. It has attracted many researchers because of its significant less training time, as it selects hidden node parameters randomly and uses inverse method that helps to get learning in one-step. The structural simplicity, computational efficiency, and advanced learning of RVFLN motivated us to investigate the efficiency of RVFLN-based forecasting. From the literature it can be seen though RVFLN model is successfully used in solving various problems, its application to financial data is scarce. Also, random initialization of hidden node parameters may degrade RVFLN performance. AEFA is a recently proposed optimization strategy inspired by the electrostatic force principle. It has shown quite acceptable approximation capability in solving realistic problems as discussed in the next section. However, it is suffering from poor exploitation ability which needs to be improved.

The following insights are drawn from the aforementioned discussions.

- Cryptocurrency prices fluctuate frequently and it is hard to explain the behavior of such price movements with conventional methodologies.
- There is still lack of sophisticated intelligent methods for cryptocurrency movement forecasting and need to be explored.
- Being a low complex ANN, RVFLN is not yet examined in cryptocurrency forecasting.
- Being a recently proposed metaheuristic, AEFA performance is not investigated in ANN optimization and cryptocurrency prediction.

The objective of this study is to design an intelligent method for cryptocurrency prices prediction using computationally efficient RVFLN with efficient learning algorithm. We first design an improved AEFA with elitism concept called eAEFA and then used eAEFA for training RVFLN (i.e., adjusting hidden node parameters of RVFLN by eAEFA). Therefore, a hybrid forecast termed as eAEFA+RVFLN is formed and employed for extrapolation of cryptocurrencies time series. The eAEFA used for RVFLN training ensures that the training process starts from a good initial point. Here sliding window method has been used to create the train and test samples from the original data series. We took into account the closing

values of six well-known cryptocurrencies, including Ripple, Litecoin, Ethereum, Z-Cash, Lumens, and Bitcoin. The predictions are assessed through error metrics such as POCID, R^2, MAPE, RMSE, and MAE to ensure the consistency of the models. There are a few comparable approaches that are designed similarly, such as MLP, SVR, ARIMA, long short-term memory (LSTM), basic RVFLN, GA+RVFLN, and AEFA+RVFLN, and the performances of the methods are assessed in terms of five metrics. The unique contributions of this article are as follows.

- The elitism concept is united with basic AEFA to advance its performance and convergence rate thus, suggesting an elitist AEFA (eAEFA) learning algorithm.
- The eAEFA is used to adjust the hidden node parameters of RVFLN and thus, improves its performance.
- The hybrid eAEFA+RVFLN is applied on prediction of next day prices of six fast budding cryptocurrencies.
- To determine the supremacy of the suggested eAEFA+RVFLN forecasts, extensive simulation studies, comparison analysis, and statistical tests are carried out.

The article is structured into seven major portions. Related articles are discussed in Section II. The methods and materials are presented in Section III, benchmark function optimization with eAEFA is discussed in Section IV, experimental outcomes are analyzed in Section V, statistical test are carried out in Section VI, concluding remarks are drawn in section VII, and potentially useful references are listed at the end.

II. RELATED STUDIES

This section discusses some of the related articles. According to the literature, various soft computing models have been successfully employed for market direction prediction [15], [16]. Inspired by the stock market prediction works; different soft computing approaches like long short-term memory neural networks (LSTM) [11], MLP [17], Random walk theory [18], SVR [19], etc. have been recommended in the literature to forecast cryptocurrencies indices. Also, for capturing threats in Cryptocurrency recurrent neural network has been [20]. In the literature, it can be seen that many researches have been conducted using different statistical approaches as well as machine learning approaches like ANN, on Bitcoin only [21], [22], [23]. Price prediction of Litecoin and Monero using reinforcement learning approach integrated with blockchain framework is conducted by authors in [24]. The study claimed to achieved better accuracy, however, it is lacking with a thorough comparative study and considering prediction of other developing digital currencies. Bitcoin and Ethereum prices are forecasted using sparrow search algorithm optimized extreme learning machine in [25]. Unlike regular stock market, cryptocurrency can be traded any time rather than specific trading time and the continuous information can affect its price instantly [26]. The complexity in accurate prediction of cryptocurrency prices is well documented in several literatures [27], [28]. Using different market and sentiments variables, Bitcoin price has been analyzed [29]. Bitcoin price forecasting methods are suggested by the authors using statistical and neural network-based methods [30], [31]. Several machines learning-based forecasting methods [32], [33], Bayesian time varying volatility model [33], random forest and gradient boosting approach [34], neural network [35], and convolutional neural networks [36] are implemented to explore and forecast cryptocurrency data.

Higher order ANNs are good enough to avoid the drawbacks associated with ANN at the same time capable of keeping all the advantages that ANNs have [37], [38], and [39]. Here the network consists of higher-order functional units which helps to replace multiple layers of ANN by a single layer. It counts the correlation among input elements which helps to get non-linear discriminant function that is useful for non-linear classification of data. RVFLN is a type of higher order ANN where to handle the non-linearity auto enhancement of input nodes is incorporated by random vector enhancements [40]. Thus, by adding functional expansions in the input layer this network efficiently replaces multiple hidden layers. According to literature RVFLN has shown significant advances in learning with proper expansion functions [40].

RVFLN models are used in a variety of fields, including the prediction of solar power [41], the classification of biological data [42], the prediction of crude oil prices [43], the forecasting of electric load [44], and the prediction of disease in the healthcare industry [45]. Combination of RVFLN with other data mining methods found success in different areas of research. Combining cascade AdaBoost detector and RVFLN, a more accurate pedestrian detection system is proposed by the authors in [46]. Combining convolutional neural network with RVFLN, a CRVFLN is designed for visual tracking [47]. A semi-supervised RVFLN is proposed by Scardapane et al. combining transudative learning theory with RVFLN [48]. To estimate the particle size in blast furnaces grinding processes, RVFLN with least-squares method is used by Dai et al. [49]. A Successive projections algorithm is used to create selective ensembles of RVFLN by Mesquita et al. [50]. In an iterative training procedure by randomly dropping out sets of connections, the ensemble RVFLN could perform better than other as proposed by authors in [51]. Ensemble empirical mode decomposition with RVFLN is used for the electricity load demand forecasting [52]. A hybrid of discrete wavelet transforms, empirical mode decomposition, and RVFLN with incremental learning approaches is used to forecast short-term electric load [53].

Non-derivative optimization techniques derived from nature have been employed in the past years. The nature-inspired algorithms are utilized to address a variety of complicated real-world issues [54]. However, due

to their numerous fine-tuned parameters, sluggish rate of convergence, lengthy execution times, etc., the applicability of these nature-inspired approaches is constrained. Evolutionary optimization techniques like GA, ACO, PSO, and DE, which require less parameters, are more effective methods [55]. There has not yet been discovered a single strategy that can tackle all types of current issues; thus, researchers have been continuously working to improve the existing approaches by either enhancing or hybridizing existing algorithms [37], [39], [56]. Recently, AEFA, an optimization strategy inspired by the electrostatic force principle, was proposed [57]. Based on the significant correlation between charged particles, i.e., the force of attraction and repulsion in an electric field, the learning capacity and convergence rate of AEFA have been observed in the base article.

III. METHODS AND MATERIALS

This section presents the baseline methods, design of eAEFA training algorithm, proposed eAEFA+RVFLN forecasting method and a summary of cryptocurrencies data used for experimentation.

A. BASIC RVFLN

RVFLN avoids the need of multiple hidden layers using functional expansions in the input layer. In this case, the customary hidden layer is shifted to the input layer. The non-linear expansion of the input nodes helps to achieve the non-linearity associated with any problem. So, selection of a suitable non-linear expansion function is very important aspect of RVFLN. The training process is explained as follows. Consider the training set be $X = \{(x(t), z(t)) | x(t) \in R^n, z(t) \in R, \forall 1 \leq t \leq \mathrm{N}\}$. Here we have considered a neural network having $n$number of input units and one output unit. Let's assume m number of nodes in hidden layer. The total number of inputs for the output node is $n+m$ since both the hidden layers and the input layer operate as input layers in this situation. In RVFLN, only computation is necessary to obtain the output layer weights, i.e., βs, as the weights are arbitrarily allocated for the link between the input and hidden layer and remain fixed during the training phase.

A mapping between the input layer and the hidden layer serves as higher order neurons in RVFLN. Let $v_j(t)$ represents the output of local induced field on node j at t^{th} iteration. So, for any node j in the hidden layer

$$v_j(t) = \sum_{k=0}^{n} \omega_{jk}(t)\, x_k(t) \tag{1}$$

Here $\omega_{jk}(t)$ is the weight distributed randomly from a uniform distribution $[-u, u]$ for $j = 1tom, k = 1ton$ for the kth input neuron to the j^{th} node. The weight ω_{j0}(t) (for fixed input $x_0 = +1$) is the bias b_j applied to neuron j. The bias $\omega_{j0}(1)$ are taken from $[0, u]$, u is a positive constant, in this case. Throughout the process, these weights are fixed, i.e., $\omega_{jk}(1)$ is assigned randomly and $\omega_{jk}(1) = \omega_{jk}(2) = \ldots = \omega_{jk}(N)$ for $j = 1\ to\ m, k = 0\ to\ n$.

At iteration t, the hidden neuron j will produce the value $y_j(t)$ which was calculated using the formula in Eq. 2.

$$y_j(t) = \varphi_j(v_j(t)) \tag{2}$$

Here, φ_j is the activation function at node j. For each hidden node $j = 1\ to\ m$, we have to calculate y_j. Now the net input that will pass to output layer is the combination of the original input set along with the calculated hidden layer outputs. This extended input set is $I = [x_1(t), x_2(t), \ldots, x_n(t), y_1(t), y_2(t), \ldots, y_m(t)]$. Consequently, the output neuron's net input will be

$$\sum_{l=0}^{n+m} \beta_l(t)\, I_l(t). \tag{3}$$

Here, $\beta_l, l = 0, 1, \ldots, n+m$; is the weight that any input neuron l assigns to the output neuron, and β_0 is the bias that the output layer node receives. The output signal $\hat{z}(t)$ occurring at the output in iteration t is determined as in Eq.4. if the output neuron has a linear transformation function.

$$\hat{z}(t) = \left(\sum_{l=0}^{n+m} \beta_l I_l(t)\right) \tag{4}$$

The basic RVFLN architecture is depicted in Figure 1. Let D and S represent the matrix of the characteristics that every training data samples will use as inputs and their anticipated results. So,

$$D = \begin{bmatrix} I_0(1), I_1(1), \ldots I_{n+m}(1) \\ I_0(2), I_1(2), \ldots I_{n+m}(2) \\ \vdots \\ \vdots \\ I_0(t), I_1(t), \ldots I_{n+m}(t) \\ \vdots \\ \vdots \\ I_0(N), I_1(N), \ldots I_{n+m}(N) \end{bmatrix} \tag{5}$$

$$\mathrm{S} = [z(1), z(2), \ldots, z(N)]^{\mathrm{T}} \tag{6}$$

D^T be the transpose of the extended input matrix D and B be the weight matrix containing β s i.e., $B = [\beta_0, \beta_1, \ldots, \beta_l, \ldots, \beta_{n+m}]^T$. If the input matrix is invertible, closed-form solutions can be used to derive the output layer weights in a single step. The values of D and S are extracted from equation 5 and 6 and are applied in equation 7.

$$D^T B = S, \quad \text{so } B = D^{-1} S \tag{7}$$

When the system of equations is inconsistent in nature so, finding inverse directly is impossible Using any pseudo-inverse techniques, we can achieve this. In such scenario we can calculate B as in Eq.8.

$$B = (D^T D)^{-1} D^T S \tag{8}$$

Here to get better solution Moore-Penrose generalized inverse also can be used to calculate matrix inverse. Once the weight matrix has been appropriately constructed, we may use Eq. 4 to forecast the test results. Usually, the number

of input samples N is large enough compared to m number of hidden neuron units, i.e., $N \geq m$. Eq. 8 indicates that there are more equations than unknowns. As a result, generalized inverse matrices are used to solve the over-determined equation system. The obtained result assures a unique solution. Transposing a matrix, multiplying a matrix, and inverting a matrix are all calculations needed in the aforementioned Eq. 8 technique used to determine the inverse. Only extra storage space is needed to store these matrixes in this non-iterative RVFLN.

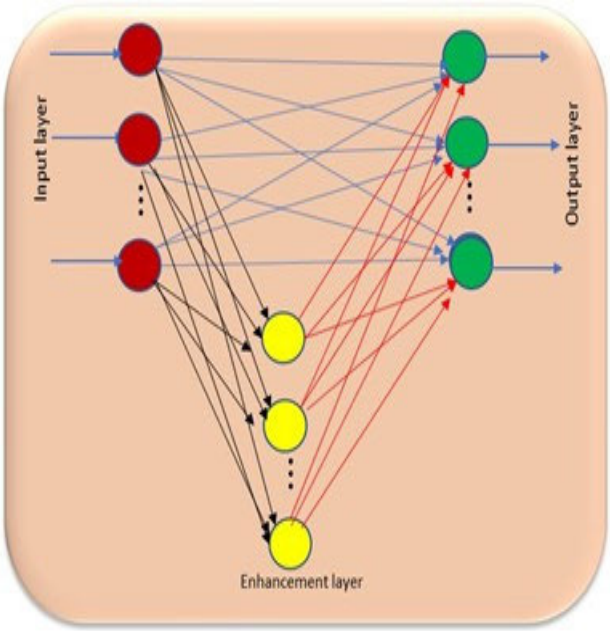

FIGURE 1. RVFLN architecture.

B. ARIMA

ARIMA models are among the widely used statistical models in financial forecasting. These are commonly referred as Box-Jenkins models and based on the hypothesis that allied time series can be generated from a linear combination of predefined number of past observations and addition of a noise term, mathematically represented as:

$$\emptyset\,(S)\,(1-S)^{d}\,(y_t) = \theta\,(S)\,\varepsilon_t \tag{9}$$

In Eq. 9, $\emptyset\,(S) = 1 - \sum_{i=1}^{p} \emptyset_i S^i$, $\boldsymbol{\theta}\,(\boldsymbol{S}) = \mathbf{1} + \sum_{j=1}^{q} \boldsymbol{\theta_j S^j}$. The parameters p, q, and d represent the number of autoregressive, the moving average terms, and the degree of differencing respectively. The term $\boldsymbol{\varepsilon_t}$ is the random error term and $\boldsymbol{y_t}$ represents the actual observations. The random error term basically satisfies the $\boldsymbol{i.i.d}$ property. Generally, these models are referred as $ARIMA(p, d, q)$. The appropriate parameters can be determined following the Box-Jenkins model build specifications.

C. SVR

SVR is a supervised learning method used for regression analysis problems. The basic formulation of SVR is as follows. Given training data $\boldsymbol{X} = \{(\boldsymbol{x_1}, \boldsymbol{y_1}), (\boldsymbol{x_2}, \boldsymbol{y_2}), \cdots, (\boldsymbol{x_n}, \boldsymbol{y_n})\}$ with $\boldsymbol{x_i} \in \boldsymbol{R^d}$ as input and $\boldsymbol{y_i} \in \boldsymbol{R}$ as corresponding output, SVR aims to find two parameters $\boldsymbol{\omega_0} \in \boldsymbol{R}$ and $\boldsymbol{\omega} \in \boldsymbol{R^d}$ by solving the constraints:

$$\begin{aligned} &\boldsymbol{min}\,\frac{1}{2}\,\|\boldsymbol{\omega}\|^2 + \boldsymbol{C}\left(\sum_{i=1}^{n} \delta_i^+ + \sum_{i=1}^{n} \delta_i^-\right) \\ &\boldsymbol{subject\ to}\ \begin{cases} (\boldsymbol{\omega}.\boldsymbol{x_i}) + \boldsymbol{\omega_0} - \boldsymbol{y_i} \leq \boldsymbol{\epsilon} + \boldsymbol{\delta_i^-} \\ \boldsymbol{y_i} - (\boldsymbol{\omega}.\boldsymbol{x_i}) - \boldsymbol{\omega_0} \leq \boldsymbol{\epsilon} + \boldsymbol{\delta_i^+} \\ \boldsymbol{\delta_i^+}, \boldsymbol{\delta_i^-} \geq \mathbf{0} \end{cases} \end{aligned} \tag{10}$$

where, $\boldsymbol{\delta_i}$ is the slack variable, C and $\boldsymbol{\epsilon}$ are input parameters greater than 0. The solution can be obtained as follows.

$$\boldsymbol{\omega} = \sum_{j=1}^{n} \left(\boldsymbol{\beta_i^-} - \boldsymbol{\beta_i^+}\right) \boldsymbol{x_i} \tag{11}$$

$$\boldsymbol{\omega_0} = \boldsymbol{y_i} - \boldsymbol{\omega}^{\boldsymbol{T}} \cdot \boldsymbol{x_i} + \boldsymbol{\epsilon} \tag{12}$$

where, $\boldsymbol{\beta_i^-}$ and $\boldsymbol{\beta_i^+}$ are the two Lagrangian parameters obtained through solving the convex quadratic programming. Now the regression function is obtained as in Eq. 13.

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{\omega}^{\boldsymbol{T}} \cdot \boldsymbol{x} + \boldsymbol{\omega_0} \tag{13}$$

D. LSTM

LSTM is a type of recurrent neural network with feedback connections and can learn long term dependencies in the data. It is most suitable for sequential data analysis such as video, speech, and dynamic time series. A typical LSTM consists a *cell* that remembers values over arbitrary time intervals and three *gates* which regulate the flow of information into/out of the cell. New information can be added or updated through the *input gate*. Irrelevant information is removed through the *forget gate*. The updated information is then passes by the *output gate* to the following LSTM cell. A gate has resemblance with a series of matrix operations that encompass discrete individual weights. The LSTM operations can be illustrated as follows:

$$f_t = Sigmoid\left(W_f X_t + U_f h_{t-1} + b_f\right) \tag{14a}$$

$$i_t = Sigmoid\left(W_i X_t + U_i h_{t-1} + b_i\right) \tag{14b}$$

$$o_t = Sigmoid\left(W_o X_t + U_o h_{t-1} + b_o\right) \tag{14c}$$

$$g_t = Tanh\left(W_c X_t + U_c h_{t-1} + b_c\right) \tag{14d}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{14e}$$

$$h_t = o_t \odot tanh(c_t) \tag{14f}$$

where, $\boldsymbol{X_t}$ is an input vector at time stamp t, W and b are the weights and bias respective layer. The cell state vector and hidden state vector are represented by $\boldsymbol{h_t}$ and $\boldsymbol{c_t}$ respectively. The input feature dimension is d and hidden unit width is h. The operator $\odot$ stands for element-wise product.

E. MLP

MLPs are widely implemented topologies applied to various fields of researches. Consider a feed forward MLP of $nm-1$ architecture where, n is input layer size, m is hidden layer size and single output neuron. Consider linear activations at input layer and sigmoid activations at hidden and output units. With X_i as the *ith* input vector, V_{ij} as the weight vector between i^{th} input neuron and j^{th} hidden neuron, each hidden unit computes as follows.

$$z_j = Sigmoid\left(bias_j + \sum_{i=1}^{n} V_{ij} * X_i\right) \tag{15}$$

The hidden nodes capture non-linear relationships among variables. The final output y_i for *ith* input vector at the output unit is computed using weight vector W_j as follows

$$y_i = Sigmoid\left(bias_0 + \sum_{j=1}^{m} W_j * z_j\right) \tag{16}$$

This output is compared with the true output and error is calculated as in Eq. 17.

$$error_i = \frac{1}{2}\sum_i (true_i - y_i)^2 \tag{17}$$

The mean error from all training patterns is calculated and propagated back to train the MLP. The parameters are adjusted by the gradient descent rule. Because of the gradient descent learning, few issues may arise such as poor convergence rate, local optima etc. which may affect the predictability.

F. BASIC AEFA

In this section we describe the optimization process of basic AEFA. It mimics a charged particle as an agent in the search space and the strength of such agent can be measured in terms of charges it possesses. A mass of such charged particles floats in a search domain with the help of electrostatic force of attraction/repulsion exist among them. Particles can interact among themselves through charges they owned. The positions of these charges are measured as potential solutions for a problem undertaken. The force of attraction is only considered in the basic AEFA, which means that all the particles associated with lower charges are attracted towards a particle owning highest charge, called as best particle or individual. The position of i^{th} charged particle (X_i) at time t is represented as in (18).

$$X_i(t) = \left(X_i^1, X_i^2, X_i^3, \cdots, X_i^D\right), \quad i = 1, 2, 3, \cdots, N \text{ and } d = 1, 2, 3, \cdots, D \tag{18}$$

where N and D is the total number of charged particles and total number of parameters (dimension) respectively. The position of i^{th} particle at time $(t+1)$ is updated as in Eq. 19, when it achieves the best fitness value.

$$P_i^d(t+1) = \begin{cases} X_i^d(t+1) & if\ fitness(X_i(t+1)) \le fitness(P_i(t)) \\ P_i^d(t) & if\ fitness(X_i(t+1)) > fitness(P_i(t)) \end{cases} \tag{19}$$

The charge associated with i^{th} particle ($Q_i(t)$) at time t is represented by Eq (20).

$$Q_i(t) = \frac{q_i(t)}{\sum_{i=1}^{N} q_i(t)} \quad i = 1, 2, \cdots, N \tag{20}$$

where $q_i(t)$ is a suitable charge function and calculated as in Eq (21) using the best fit and worst fit particles in the search space.

$$q_i(t) = exp^{\left(\frac{fitness_i(t) - fitness_{worst}(t)}{fitness_{best}(t) - fitness_{worst}(t)}\right)} \tag{21}$$

The force $F_{ij}^d(t)$ experienced at i^{th} particle holding charge $Q_i(t)$ due to j^{th} particle holding charge $Q_j(t)$ is defined as in Eq. 22.

$$F_{ij}^d(t) = K(t)\frac{Q_i(t) \cdot Q_j(t) \cdot (P_j^d(t) - X_i^d(t))}{\left\|X_i(t) - X_j(t)\right\|^2 + \varepsilon} \tag{22}$$

where, $K(t)$ is the Coulomb's constant calculated in terms of current iteration and maximum iteration as in Eq (23) and ε is a small positive constant.

$$K(t) = K_0 \cdot exp^{\left(-\alpha \frac{iteration}{max.iteration}\right)} \tag{23}$$

The value of parameter $\alpha = 30$ and $K_0 = 500$. The bigger initial value of K_0 helps in better exploration in the search process and gradually decreases through iterations to regulate the accuracy. The resultant electrostatic force F_i^d acting on i^{th} particle at time t can be calculated as in Eq (24) and the electric field is calculated as in Eq (25).

$$F_i^d(t) = \sum rand \cdot F_{ij}^d(t), \quad j = 1, 2, \cdots, N \text{ and } i \neq j \tag{24}$$

$$E_i^d(t) = \frac{F_i^d(t)}{Q_i(t)} \tag{25}$$

As per Newton's law of motion, the acceleration $a_i^d(t)$ of i^{th} charged particle having unit mass $M_i(t)$ at time t is computed as in Eq (26).

$$a_i^d(t) = \frac{Q_i(t) \cdot E_i^d(t)}{M_i(t)} \tag{26}$$

The velocity and position of i^{th} charged particle at time (t +1) are updated according to Eq (27) and Eq (28) respectively.

$$V_i^d(t+1) = rand_i * V_i^d(t) + acceleration_i^d(t) \tag{27}$$

$$X_i^d(t+1) = X_i^d(t) + V_i^d(t+1) \tag{28}$$

A particle associated with maximum quantity of charges can be considered as a best individual. This individual particle attracts other particles having lesser charge and voyages in the search domain.

G. eAEFA

As was previously said, a charged particle with the highest charge among them all is deemed to be the best individual, draws other charged particles with a lower charge, and proceeds through the search field. Elite solutions, on the other hand, can make up a small percentage of the good ones in the preceding step. In an elitism mechanism, the elite solutions are immediately and unchanged passed on to the following generation. Instead of inferior solutions, use elite ones. The elite identified in the generation before replace any generation's worst solutions. Numerous swarm and evolutionary algorithms are used in conjunction with the elitism mechanism, which replaces the poorest solutions with the best ones after each generation [58].

We use the elitism method to accelerate AEFA's convergence. Figure 2 displays the eAEFA algorithm. This ensures that the system always initiates the best candidates during the optimization process. A randomly chosen initial population of solutions is used to begin the process. The initial bias and weight of a potential ANN are represented by an entity drawn from the population. The ANN model's fitness is assessed once this population and the input samples have been fed into it. An elite group of solutions is chosen based on fitness. The regular operators of AEFA take care of the rest of the population. When the current generation is finished, the modified and original solutions are contrasted, and the winning one is used going forward. Here, the elite solutions take the place of the inferior ones, and the cycle repeats itself for the following generation. The elite solutions are transmitted in this way through succeeding generations. Finally, the best answer is held upon and put to test.

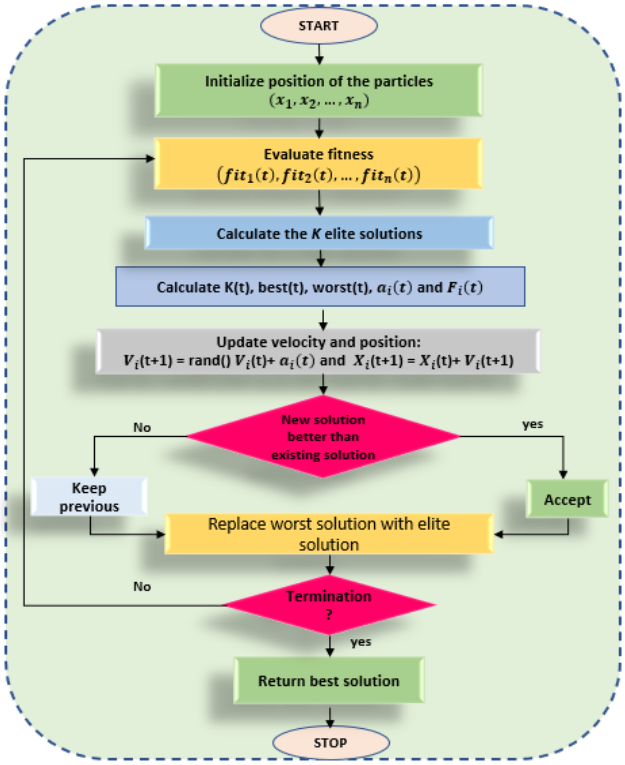

FIGURE 2. eAEFA optimization process flow.

H. eAEFA+ RVFLN FORECASTING METHOD

Here, we outline the method of input selection, normalization, and eAEFA+RVFLN forecasting procedure.

1) INPUT SELECTION

The digital currencies that have been gathered from websites are regarded as financial time series. We then used a moving window from the original series to build the input patterns for the model. Fig. 3 depicts the input pattern generating process. Instead of choosing closing prices from all previous days to feed the network, we have used a sliding window with a predetermined width as training input in this case. As a substitute of selecting all observed data, we make decisions based on current and a few recent data points. The window receives new data and discards the oldest with each movement. In this manner, the moving window moves along the index's series, and the width of the window is calculated. For financial data, we cannot select random samples and assign them to either the test set or the train set, in contrast to the cross-validation approach, which is a normal procedure for ordinary time series. This is because it is absurd to use values from the future to forecast values in the past. As a result, using the sliding window approach, we first projected for the later data using a subset of the data for training, then we verified the forecast's accuracy. Following that, the identical forecasted data point is incorporated in the next training dataset, and following data points are forecasted.

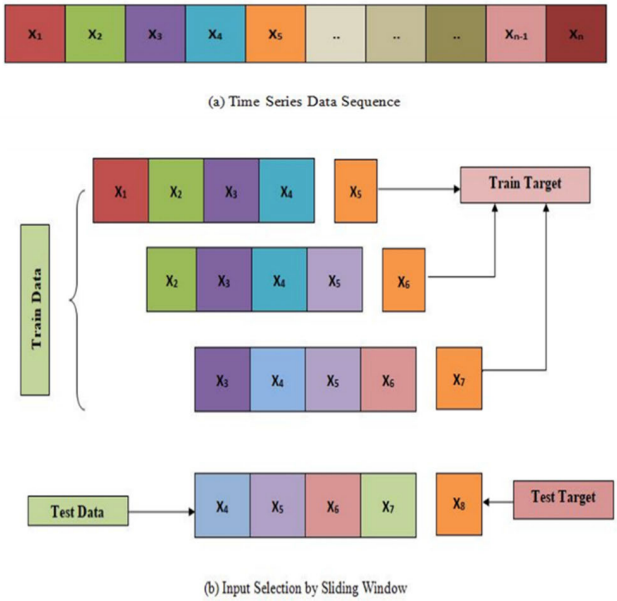

FIGURE 3. Input selection from original time series using sliding window method.

TABLE 1. Cryptocurrency series for experimentation.

S. No.	Crypto data series	Short name	Total no. of data points	Total no. of input patterns generated by rolling window
1	Bitcoin	BTC	1363	1352
2	Litecoin,	LTC	1244	1233
3	Ethereum	ETH	1244	1233
4	Z-cash	ZEC	782	771
5	Lumens	XLM	1075	1064
6	Ripple	XRP	1102	1091

2) NORMALIZATION

The patterns are normalized in order to scale the data for each input characteristic into a consistent range. We employed the tanh estimator approach, as shown in Eq. 29, to normalize time series data. The moving window's mean and standard deviation are shown here by μ and σ, respectively.

$$\hat{x} = 0.5 * \left(tanh \left(\frac{0.01 * (x - \mu)}{\sigma} \right) + 1 \right) \tag{29}$$

3) FORECASTING

Following normalization, the patterns are successively fed into the eAEFA+RVFLN forecast along with the randomly initialized weight and biases for the hidden layer. Given a series of $k$recent past prices and current day price $\{Price_{current-k}, \cdots, Price_{current-1}.Price_{current}$, the process predicts a price for nest day, i.e., $Price_{next}$ as follows.

$$Price_{next} = eAEFA + RVFLN\ (Price_{current-k}, \cdots, Price_{current-1}.Price_{current}) \tag{30}$$

eAEFA+RVFLN compute the output layer weights as of Eq.7. Only one defendant variable exists, hence there is only one neuron in the output layer (the model is only predicting one value). The fitness is defined as the magnitude of the discrepancy between the actual output and the estimated output at the output layer. The model is then trained using eAEFA as described in Section III-G, using the weights determined in RVFLN as the initial population. Algorithm 1 gives a description of the fundamental RVFLN algorithm and Algorithm 2 shows the high-level RVFLN-based forecasting. The schematic diagram of eAEFA+RVFLN based cryptocurrency forecasting is depicted in Figure 4. As in Eq.31, the model estimated an output (y) at the output layer. The magnitude of the estimation's deviation from the target (y) is calculated as an error in Eq. 32. The accuracy of the model's forecast is increased by having a low error.

$$\hat{y} = \sum_{i=1}^{m} (\beta_i * x_i + b) \tag{31}$$

$$error = abs\left(y - \hat{y}\right) \tag{32}$$

Each particle of eAEFA represents a set of hidden layer parameters (weights, biases, and number of hidden neurons). The fitness function for the learning process is shown in Eq. 33.

$$Fitness = \frac{\sum_{j=1}^{N} \sum_{i=1}^{m} \left| \left(\beta_i \times f\left(w_i x_j + bias_i\right) - target_j\right) \right|}{m \times N} \tag{33}$$

Algorithm 1 Evaluate RVFLN ($\boldsymbol{X}, \boldsymbol{P}$)

Input: Training set $X = \{a_i, t_i\}, i = 1, .., N$, non-linear activation functions $\emptyset$, number of enhancement nodes$m$, and population $\{P = W_{jk}\}$.
Output: Predictionerror

Begin

1. For an input vector be $A_{n\times 1} \in X$, with target set $T_{t\times 1}$
2. Initialize random weights of hidden layer $W_{n\times m}$
3. Compute weighted sum $Y = W^T A$
4. Compute $Y^* = \emptyset\,(Y)$
5. Construct extended matrix $B_{(m+n)\times 1} = [Y^* A]$
6. Compute output weight matrix $Z_{(m+n)\times t}$ using least square norm solution.
7. Compute $Z^T = TB^+$ and $T^* = Z^T B$.
8. Compute $error = \|T^* - T\|$
9. return *error*

End

Algorithm 2 eAEFA+RVFLN Forecasting Method

Step 1. Form *TrainData* and *TestData* using sliding window
Step 2. Normalize *TrainData* and *TestData*
Step 3. /*Train RVFLN with *TrainData**/
While (*Data not* exhausted)

$Fitness = EvaluateRVFLN(TrainData, P)$

Sort the solutions according to fitness and identify k elite solutions.
Apply eAEFA and update population.
Obtain optimal population P
end While
Step 4. /*Test RVFLN with *TrainData**/

$Error = EvaluateRVFLN(TestData, P)$

I. CRYPTOCURRENCY DATA

The data on cryptocurrencies used in this study, their sources, and a summary description were all described in this section. To determine the data's stationarity and non-linearity, some tests are run on it. Six cryptocurrency indices, including Bitcoin, Litecoin, Ethereum, ZEC, XLM, and Ripple, are used to simulate the proposed and compared models. For these six series, the closing prices were taken from "*https://www.CryptoDataDownload.com*." Table 1 displays information about the six financial series. The data was gathered between April 17, 2017, and May 6, 2021.

TABLE 2. Summary statistics from cryptocurrency series.

Statistic	BTC	LTC	ETH	ZEC	XLM	XRP
Minimum	3.1890e+03	23.0800	23.0800	1.00000	0.0317	0.1355
Mean	1.2374e+04	92.9374	92.9374	72.7968	0.1508	0.3698
Median	8600	63.9900	63.9900	61.2250	0.1002	0.3037
Variance	1.6080e+08	4.0937e+03	4.0937e+03	2.2159e+03	0.0149	0.0521
Maximum	63575	356.0400	356.0400	320.0100	0.6570	1.8347
Standard deviation	1.2681e+04	63.9822	63.9822	47.0730	0.1220	0.2283
Skewness	2.6519	1.5433	1.5433	2.4708	1.7898	3.3673
Kurtosis	9.1102	4.9659	4.9659	10.1543	5.8953	17.2210
Correlation coefficient	0.00275	-0.13106	-0.11463	0.00857	-0.0427	0.03750

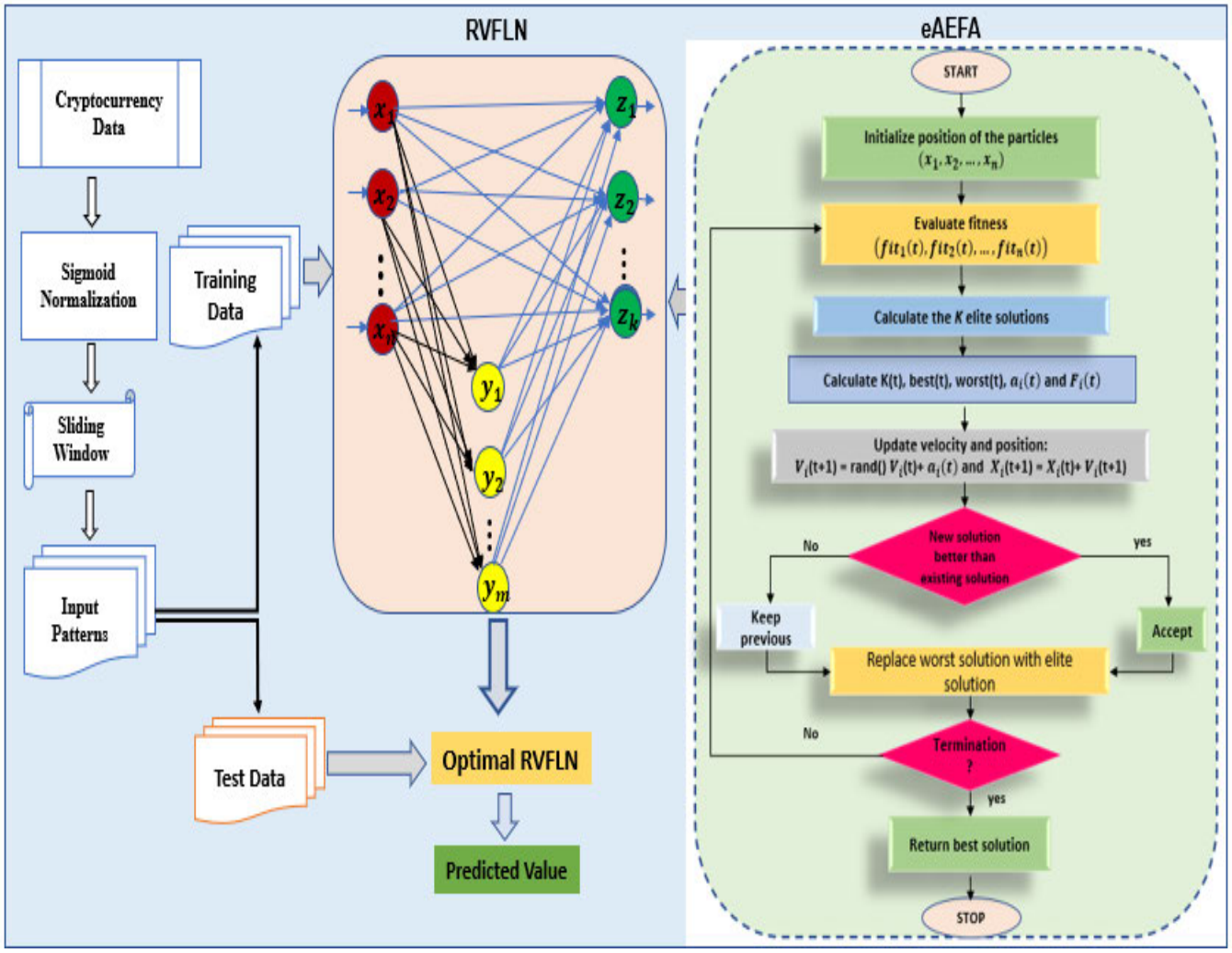

FIGURE 4. Schematic diagram of eAEFA+RVFLN based cryptocurrency price forecasting.

The descriptions of the statistical characteristics of the datasets that we examined are condensed in Table 2. The six closing price series are shown in Figure 5 - 10. All of the series contain significant volatility and standard deviation, as seen in Table 2. Positive skewness pervades every series. All series' non-zero skewness values point to asymmetry in the distributions. Since all series have larger kurtosis values, they all have large tails. The return series connections with the return prices are less strong.

To check for series stationarity, we employed two well-known tests such as Phillips-Perron (PP) test and Augmented Dickey-Fuller (ADF). The statistics from the series stationarity check are presented in Table 3. Stationarity of a financial dataset is a vital feature to make reports about its

TABLE 3. Statistics from PP test and ADF test for non-stationarity in the datasets.

Data	Indicator	ADF test		PP test	
		Level	1st Difference	Level	1st Difference
BTC	*t-stat.* (Prob.)	-2.364 (0.3203)	-40.5278 (0.0001)	-3.4682 (0.3913)	-52.1094 (0.0000)
	t-critical (5%)	-3.3358	-3.4337	-3.1308	-3.04437
LTC	*t-stat.* (Prob.)	-2.3758 (0.4531)	-41.3760 (0.0000)	-3.7302 (0.3955)	-60.7765 (0.0000)
	t-critical (5%)	-3.5139	-3.4739	-3.8109	-3.5419
ETH	*t-stat.* (Prob.)	-2.5023 (0.2915)	-46.1265 (0.0001)	-3.5625 (0.345)	-65.9133 (0.0001)
	t-critical (5%)	-3.4249	-3.6439	-3.4395	-4.4378
XLM	*t-stat.* (Prob.)	-2.3880 (0.4738)	-42.3226 (0.0000)	-4.8135 (0.5563)	-64.4856 (0.0000)
	t-critical (5%)	-3.3944	-3.4876	-3.4290	-3.4039
ZEC	*t-stat.* (Prob.)	-3.1625 (0.0730)	-41.8609 (0.0000)	-4.7290 (0.0839)	-59.4765 (0.0000)
	t-critical (5%)	-3.5386	-3.1348	-3.4285	-3.4039
XRP	*t-stat.* (Prob.)	-2.5525 (0.2755)	-45.1465 (0.0001)	-3.3628 (0.3455)	-62.3384 (0.0001)
	t-critical (5%)	-3.4245	-3.7435	-3.5305	-4.4577

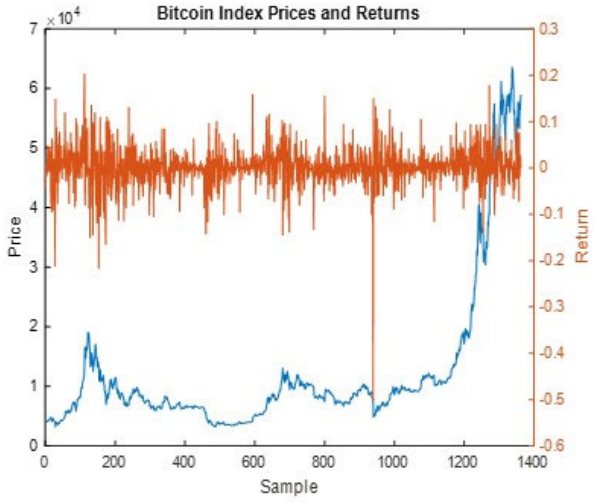

FIGURE 5. Price movement patterns of Bitcoin.

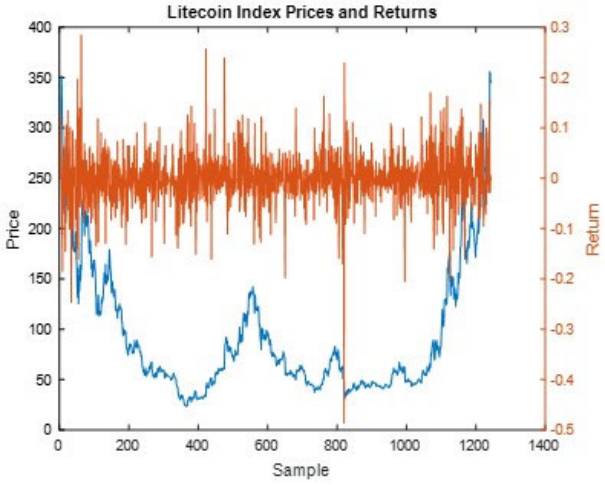

FIGURE 6. Price movement patterns of Litecoin.

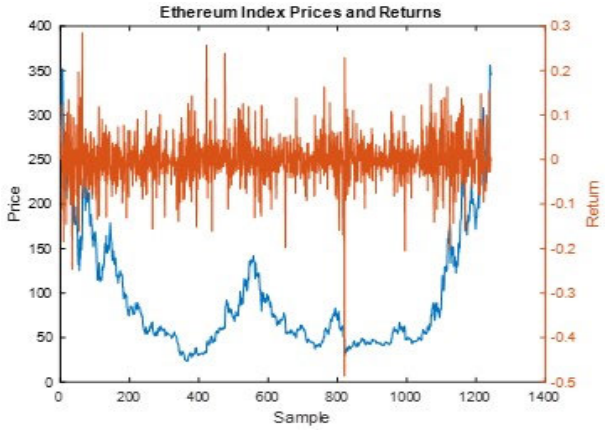

FIGURE 7. Price movement patterns of Ethereum.

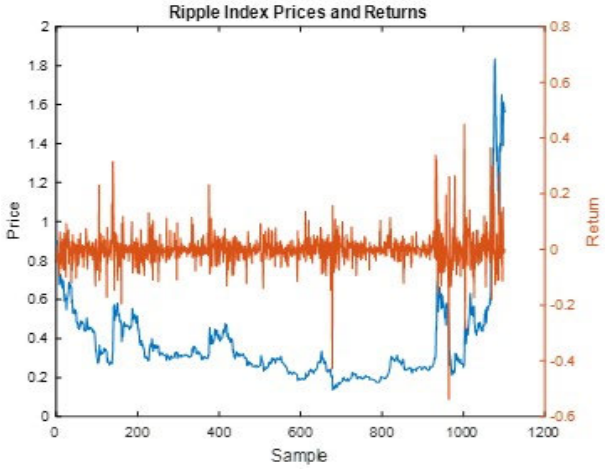

FIGURE 8. Price movement patterns of Ripple.

future. The stationarity checking process investigates presence of unit root in the dataset. The following are the null hypothesis and alternative of ADF.

H_0: $\alpha = 0$, indicates the presence of a unit root and that the series is non-stationary.

H_{alt}.: $\alpha < 0$, means there is no unit root and the series is stationary.

It is evaluated as $t_\alpha = \frac{\hat{\alpha}}{se(\hat{\alpha})}$, Where $\hat{\alpha}$ is an estimation for and $se(\hat{\alpha})$ is coefficient standard error. The calculated statistics is compared with the critical values. H_0 is rejected if

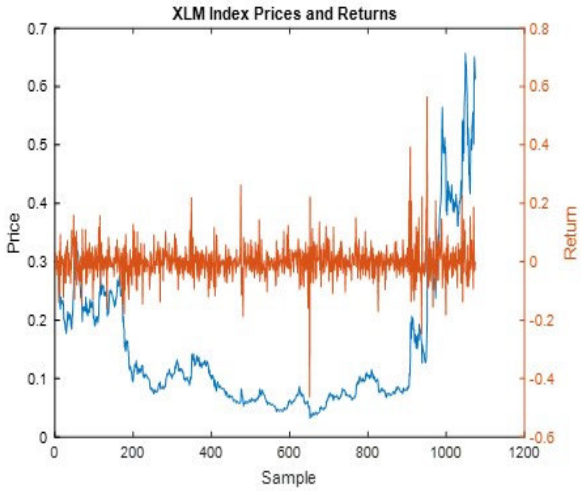

FIGURE 9. Price movement patterns of XLM.

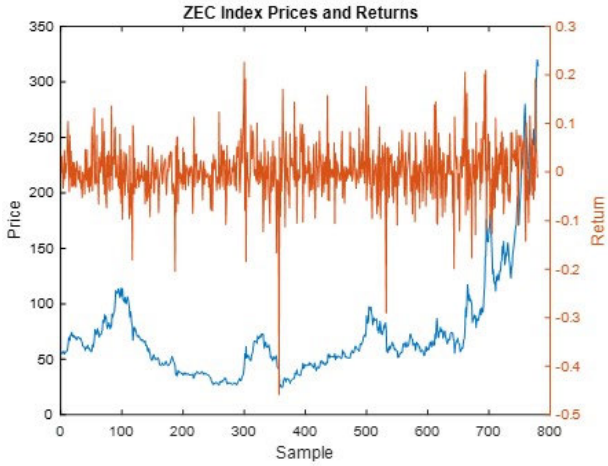

FIGURE 10. Price movement patterns of ZEC.

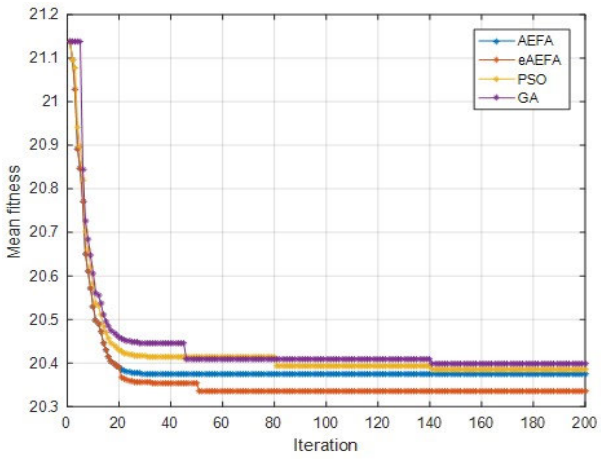

FIGURE 11. Convergence curves of comparative algorithms on Ackley's function.

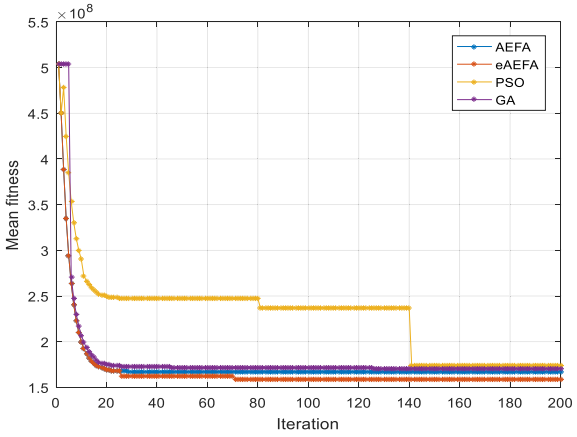

FIGURE 12. Convergence curves of comparative algorithms on Rosenbrock's function.

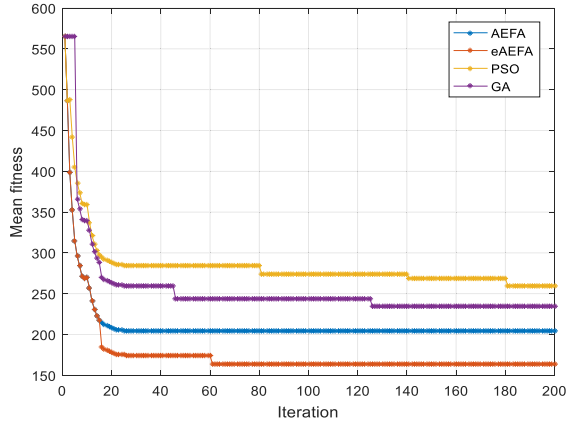

FIGURE 13. Convergence curves of comparative algorithms on Rastrigin's function.

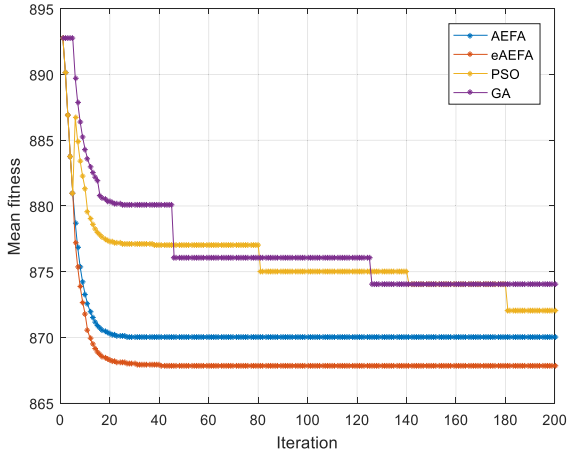

FIGURE 14. Convergence curves of comparative algorithms on Griewank's function.

t_α < critical value, saying that no unit root found and the series is stationary. The PP test is a non-parametric approach, similar to ADF, but it also acknowledges that residues are auto-correlated by an automated correction during testing. The results from Table 3 confirmed the Box-Jenkins technique by demonstrating non-stationarity in levels, stationarity

TABLE 4. Comparative results of Ackley's function.

Metric	Iteration No. = 50			
	eAEFA	AEFA	PSO	GA
best	**20.0717**	21.2643	22.4709	22.8055
worst	**21.1943**	23.0034	23.9837	24.5909
median	**20.1715**	21.1755	22.7238	22.6388
std	**0.0326**	0.1222	0.1749	0.1763
	Iteration No. = 100			
best	**20.2344**	20.3744	20.3844	20.3994
worst	**21.1236**	21.1394	21.1394	21.1594
median	**20.2344**	20.3744	20.3944	20.4094
std	**0.1076**	0.1158	0.1195	0.1241
	Iteration No. = 150			
best	**20.2351**	20.3415	20.3540	20.3004
worst	21.1942	**20.9385**	21.1300	21.1533
median	20.3306	20.3342	20.3944	**20.1477**
std	**0.1076**	0.1157	0.1195	0.1246

TABLE 5. Comparative results of Rosenbrock's function.

Metric	Iteration No. = 50			
	eAEFA	AEFA	PSO	GA
best	**1.5916e+08**	1.6915e+08	2.0891e+08	2.7222e+08
worst	**5.0575e+08**	5.1273e+08	5.6883e+08	5.3831e+08
median	**1.5879e+08**	1.6505e+08	2.4687e+08	1.7952e+08
std	3.9953e+07	**3.8448e+07**	5.2878e+07	5.1493e+07
	Iteration No. = 100			
best	**1.5916e+08**	1.6936e+08	1.7895e+08	1.7725e+08
worst	**5.0571e+08**	5.1273e+08	5.4703e+08	5.7437e+08
median	**1.5888e+08**	1.6508e+08	2.3768e+08	1.7312e+08
std	**3.6558e+07**	3.8614e+07	5.1428e+07	5.2861e+07
	Iteration No. = 150			
best	**1.5918e+08**	1.6718e+08	1.7368e+08	1.7023e+08
worst	**5.0373e+08**	**5.0373e+08**	**5.0373e+08**	**5.0373e+08**
median	**1.5918e+08**	1.6718e+08	2.3718e+08	1.7118e+08
std	3.9958e+07	**3.8616e+07**	5.2873e+07	5.0469e+07

TABLE 6. Comparative results of Rastrigin's function.

Metric	Iteration No. = 50			
	eAEFA	AEFA	PSO	GA
best	**160.1777**	204.0177	281.4628	261.3586
worst	**546.9839**	**546.9839**	590.9235	588.7700
median	**163.9138**	204.0177	278.5376	268.2839
std	46.9142	**38.9550**	39.5088	57.3688
	Iteration No. = 100			
best	**160.1777**	204.0173	266.1246	239.3755
worst	**546.9839**	**546.9839**	561.9833	568.0837
median	**163.9166**	208.0175	274.2374	254.2703
std	**36.5742**	38.9555	**36.5782**	55.9203
	Iteration No. = 150			
best	**159.0947**	204.0168	260.0277	234.0327
worst	**546.9235**	**546.9235**	564.9839	564.9839
median	**154.0725**	204.0177	274.0377	244.0527
std	46.9768	38.9550	**36.7983**	54.0339

in first differences, and support rejection of the null hypothesis that all data series are significantly non-stationary.

IV. eAEFA PERFORMANCE ON BENCHMARK FUNCTION OPTIMIZATION

The optimization capability and convergence of basic AEFA is proved by the inventors in [57]. The powerful global optimization ability of AEFA is compared with few states of the art evolutionary algorithms and established as a superior metaheuristic. We proposed eAEFA in this article and to realize the effect of elitism on AEFA learning, we first employed eAEFA for benchmark function optimization and compared its performance with basic AEFA, and two widely used optimization algorithms such as PSO and GA. The four benchmark functions are Ackley's function (Eq. 34), Rosenbrock's function (Eq. 35), Rastrigin's function (Eq. 36), and Griewank's function (Eq. 37) detailed as follows.

$$f_1(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e \quad (34)$$

$$f_2(x) = \sum_{i=1}^{D-1}\left(100\left(x_i^2 - x_{i+1}\right)^2 + (x_i - 1)^2\right) \quad (35)$$

$$f_2(x) = \sum_{i=1}^{D}\left(x_i^2 - cos(2\pi x_i) + 10\right) \quad (36)$$

$$f_2(x) = \sum_{i=1}^{D}\frac{x_i^2}{4000} - \prod_{i=1}^{D} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \quad (37)$$

TABLE 7. Comparative results of Griewank's function.

Metric	Iteration No. = 50			
	eAEFA	AEFA	PSO	GA
best	**822.3061**	844.5488	885.3260	865.3265
worst	**846.0385**	862.4735	895.7385	892.7582
median	**825.1433**	846.3615	844.6278	849.8675
std	**2.8955**	4.8515	2.8992	3.2155
	Iteration No. = 100			
best	**823.3260**	837.3677	851.4058	847.8268
worst	842.7385	**842.3380**	852.3845	892.7385
median	**830.6500**	842.7446	848.3752	847.8268
std	**2.8955**	2.8995	3.5985	3.2100
	Iteration No. = 150			
best	**823.0268**	837.8268	852.0368	844.0418
worst	**842.7385**	892.7385	892.7385	892.7385
median	**830.0268**	867.8268	855.0468	846.0618
std	**2.8651**	3.2100	2.8692	3.8213

TABLE 8. Performance metrics used to evaluate the forecasting models.

Performance measure	Computed formula	Ideal value
MAPE	$\frac{1}{N}\sum_{i=1}^{N}\frac{\lvert x_i - \hat{x}_i\rvert}{x_i} \times 100$	0
R^2	$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$, $SS_{res} = \sum_{i=1}^{N}(x_i - \hat{x}_i)^2$ and $SS_{tot} = \sum_{i=1}^{N}(x_i - \bar{X})^2$	1
POCID	$\frac{\sum_{i=1}^{N} Trend_i}{N} * 100$, $Trend_i = \begin{cases} 1, if\ (x_i - x_{i-1})(\hat{x}_i - \hat{x}_{i-1}) > 0 \\ 0, otherwise \end{cases}$	1
RMSE	$\sqrt{1/N\sum_{i=1}^{N}(x_i - \hat{x}_i)^2}$	0
MAE	$\frac{1}{N}\sum_{i=1}^{N}\lvert x_i - \hat{x}_i\rvert$	0

TABLE 9. MAPE statistics from six datasets and six prediction models.

MODEL	MAPE						Average Rank
	BTC	LTC	ETH	ZEC	XLM	XRP	
ARIMA	0.6165 (7)	0.4427 (8)	0.3675 (7)	0.6275 (8)	0.3668 (8)	0.1428 (7)	7.50
MLP	0.3065 (6)	0.1827 (7)	0.1527 (5)	0.5471 (7)	0.2572 (7)	0.0979 (6)	6.33
SVR	0.6188 (8)	0.0793 (6)	0.3801 (8)	0.3666 (6)	0.2133 (6)	0.1725 (8)	7.00
LSTM	0.0892 (4)	0.0762 (4.5)	0.1577 (6)	0.3501 (3)	0.2005 (5)	0.0511 (3)	4.25
RVFLN	0.0931 (5)	0.0762 (4.5)	0.0842 (4)	0.3625 (5)	0.1634 (4)	0.0763 (5)	4.58
GA+RVFLN	0.0857 (2)	0.0558 (2)	0.0479 (3)	0.3560 (4)	0.0775 (3)	0.0552 (4)	3.00
AEFA+RVFLN	0.0879 (3)	0.0585 (3)	0.0475 (2)	0.2374 (2)	0.0735 (1)	0.0525 (2)	2.16
eAEFA+RVFLN	0.0802 (1)	0.0260 (1)	0.0323 (1)	0.1065 (1)	0.0752 (2)	0.0236 (1)	1.16

TABLE 10. R^2 statistics from six datasets and six prediction models.

MODEL	R^2						Average Rank
	BTC	LTC	ETH	ZEC	XLM	XRP	
ARIMA	0.8044(6)	0.7638(7.5)	0.7655 (6)	0.6688 (8)	0.8600 (6)	0.7628 (6)	6.58
MLP	0.7842 (7)	0.7560 (6)	0.7463 (7)	0.7158 (6)	0.8290 (8)	0.7364 (7)	6.83
SVR	0.7588 (8)	0.9109 (5)	0.7422 (8)	0.7113 (7)	0.8544 (7)	0.7315 (8)	7.16
LSTM	0.9125 (4)	0.9385 (4)	0.8600 (5)	0.8593 (5)	0.8817 (4)	0.8602 (5)	4.50
RVFLN	0.8275 (5)	0.7638(7.5)	0.8657 (3)	0.8634 (4)	0.8628 (5)	0.8619 (4)	4.75
GA+RVFLN	0.9160 (3)	0.9525 (2)	0.9198 (2)	0.9576 (2.5)	0.9077 (2)	0.9520 (2)	2.25
AEFA+RVFLN	0.9435 (2)	0.9537 (1)	0.8615 (4)	0.9576 (2.5)	0.8841(3)	0.8864 (3)	2.58
eAEFA+RVFLN	0.9862 (1)	0.9505 (3)	0.9478 (1)	0.9735 (1)	0.9368 (1)	0.9586 (1)	1.33

TABLE 11. POCID statistics from six datasets and six prediction models.

MODEL	POCID						Average Rank
	BTC	LTC	ETH	ZEC	XLM	XRP	
ARIMA	0.7955 (7)	0.7863 (7)	0.6533 (8)	0.7805 (8)	0.7024 (8)	0.8089 (7)	7.50
MLP	0.7535 (8)	0.7193 (8)	0.7183 (6)	0.7896 (6)	0.7793 (6)	0.8482 (6)	6.66
SVR	0.9022 (6)	0.7837 (6)	0.7144 (7)	0.7806 (7)	0.7584 (7)	0.7893 (8)	6.83
LSTM	0.9535 (2)	0.9478 (3)	0.8670 (5)	0.9128 (2)	0.8316 (5)	0.8499 (5)	3.66
RVFLN	0.8623 (5)	0.8458 (4)	0.8688 (3)	0.8406 (5)	0.8395 (4)	0.8604 (4)	4.16
GA+RVFLN	0.8979 (4)	0.8004 (5)	0.8674 (4)	0.8655 (4)	0.8683 (3)	0.8852 (3)	3.83
AEFA+RVFLN	0.9098 (3)	0.9643 (1)	0.9608 (2)	0.8949 (3)	0.9077 (2)	0.9528 (1.5)	2.08
eAEFA+RVFLN	0.9868 (1)	0.9642 (2)	0.9633 (1)	0.9529 (1)	0.9859 (1)	0.9528 (1.5)	1.25

The domain dimension and search range of these functions was set to 50 and [−100 100] respectively. The learning parameters of four optimization techniques are chosen experimentally. The algorithms are iterated 200 times. The number of particles for AEFA and eAEFA was set to 50. The swarm size of PSO and chromosome number for GA are chosen as 80 and 100 respectively. The crossover and mutation probability of GA are chosen as 0.7 and 0.002 respectively. For PSO, the inertia weight, cognitive and social parameter values are set to 0.736248, 1.42374, and 1.42374

TABLE 12. RMSE statistics from six datasets and six prediction models.

MODEL	RMSE						Average Rank
	BTC	LTC	ETH	ZEC	XLM	XRP	
ARIMA	0.6352 (8)	0.3840 (8)	0.3884 (8)	0.2836 (8)	0.4752 (7)	0.4698 (8)	7.83
MLP	0.3850 (7)	0.0995 (6)	0.3695 (6)	0.0683 (5)	0.2773 (6)	0.2698 (6)	6.00
SVR	0.1895 (6)	0.1302 (7)	0.3699 (7)	0.0788 (7)	0.4789 (8)	0.2840 (7)	7.00
LSTM	0.0799 (3)	0.0988 (5)	0.0877 (5)	0.0611 (4)	0.0935 (4)	0.0697 (5)	4.33
RVFLN	0.1687 (5)	0.0984 (4)	0.0699 (4)	0.0747 (6)	0.1673 (5)	0.0689 (4)	4.66
GA+RVFLN	0.0860 (4)	0.0599 (1)	0.0669 (3)	0.0598 (3)	0.0896 (3)	0.0658 (3)	2.83
AEFA+RVFLN	0.0715 (1)	0.0649 (2)	0.0645 (1)	0.0585 (2)	0.0890 (2)	0.0584 (1)	1.50
eAEFA+RVFLN	0.0790 (2)	0.0656 (3)	0.0648 (2)	0.0548 (1)	0.0875 (1)	0.0596 (2)	1.83

TABLE 13. MAE statistics from six datasets and six prediction models.

MODEL	MAE						Average Rank
	BTC	LTC	ETH	ZEC	XLM	XRP	
ARIMA	0.3653 (8)	0.2520 (8)	0.2965 (8)	0.4386 (8)	0.4796 (8)	0.4904 (8)	8.00
MLP	0.3424 (6)	0.1675 (7)	0.1544 (6)	0.2923 (7)	0.3695 (6)	0.2865 (7)	6.50
SVR	0.2777 (7)	0.1326 (6)	0.1583 (7)	0.1048 (6)	0.3801 (7)	0.1740 (6)	6.50
LSTM	0.0937 (4)	0.0979 (5)	0.0874 (5)	0.0937 (4)	0.3066 (5)	0.0949 (5)	4.66
RVFLN	0.2653 (5)	0.0928 (4)	0.0795 (3)	0.3682 (5)	0.2875 (4)	0.0945 (4)	4.16
GA+RVFLN	0.0879 (2)	0.0693 (2)	0.0833 (4)	0.0726 (3)	0.1679 (3)	0.0709 (3)	2.83
AEFA+RVFLN	0.0898 (3)	0.0778 (3)	0.0733 (2)	0.0659 (2)	0.0937 (2)	0.0629 (1)	2.16
eAEFA+RVFLN	0.0864 (1)	0.0685 (1)	0.0675 (1)	0.0574 (1)	0.0886 (1)	0.0678 (2)	1.16

TABLE 14. Overall ranking of the forecasts considering five metrics and six prediction models.

MODEL	Error Measure					Overall rank
	MAPE	R^2	*POCID*	*RMSE*	*MAE*	
ARIMA	7.50	6.58	7.50	7.83	8.00	**7.482**
MLP	6.33	6.83	6.66	6.00	6.50	**6.464**
SVR	7.00	7.16	6.83	7.00	6.50	**6.898**
LSTM	4.25	4.50	3.66	4.33	4.66	**4.28**
RVFLN	4.58	4.75	4.16	4.66	4.16	**4.462**
GA+RVFLN	3.00	2.25	3.83	2.83	2.83	**2.948**
AEFA+RVFLN	2.16	2.58	2.08	1.50	2.16	**2.096**
eAEFA+RVFLN	1.16	1.33	1.25	1.83	1.16	**1.346**

respectively referring to literature. The convergence graph of four optimization methods from four benchmark functions are illustrated by Figures 11 – 14. The mean fitness values of the four objective functions are recorded at 50, 100, and 150 number of iterations and enlisted in Table 4 - 7.

From these convergence plots, it can be observed that, both eAEFA and basic AEFA converged rapidly and more accurately than PSO and GA. The eAEFA converged slightly better than AEFA and maintained well balance between exploration and exploitation. In each row of Table 4 – 7, the best fitness values are presented in boldface. It is found that eAEFA and AEFA obtained best values within 50 iterations while PSO and GA reached their best values with more than hundred iterations. In comparison with three algorithms, eAEFA performed significantly better than others. The GA yielded superior results compared to PSO.

V. EXPERIMENTAL OUTCOMES FROM CRYPTOCURRENCY FORECASTING

Six data series described in the preceding section are used in a variety of experiments to test the suggested and compared models. Using the identical input patterns, seven comparable models such as ARIMA, MLP, SVR, LSTM, RVFLN, GA+RVFLN, and AEFA+RVFLN are built and assessed. The training and test patterns are normalized and given independently to the eight models following the windowing method. An error signal is a model estimated output that deviates from the desired result. The accuracy of a model is

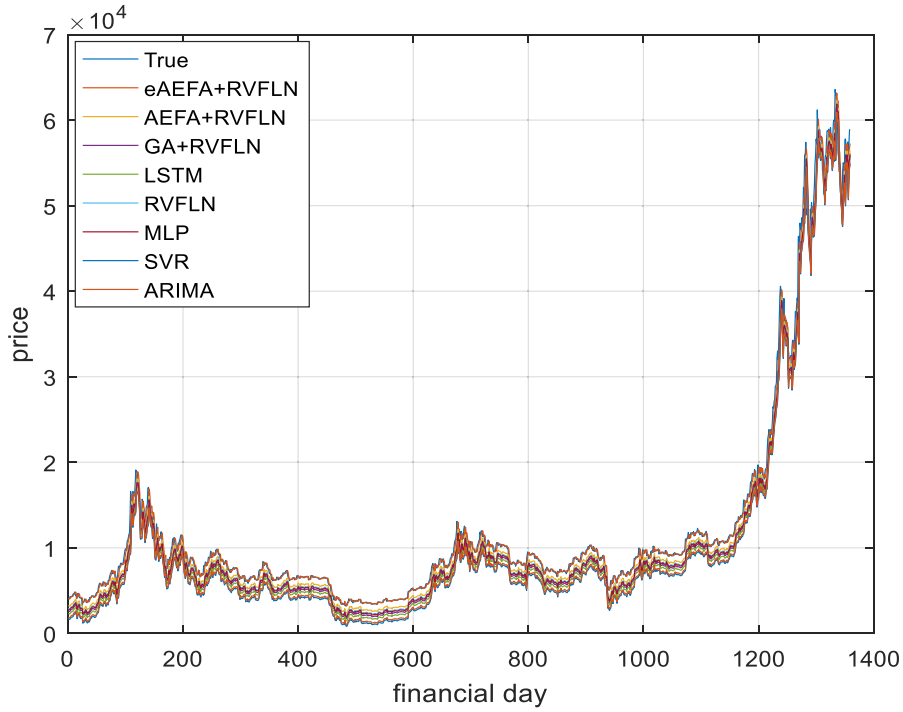

FIGURE 15. Actual v/s estimated from BTC forecasting.

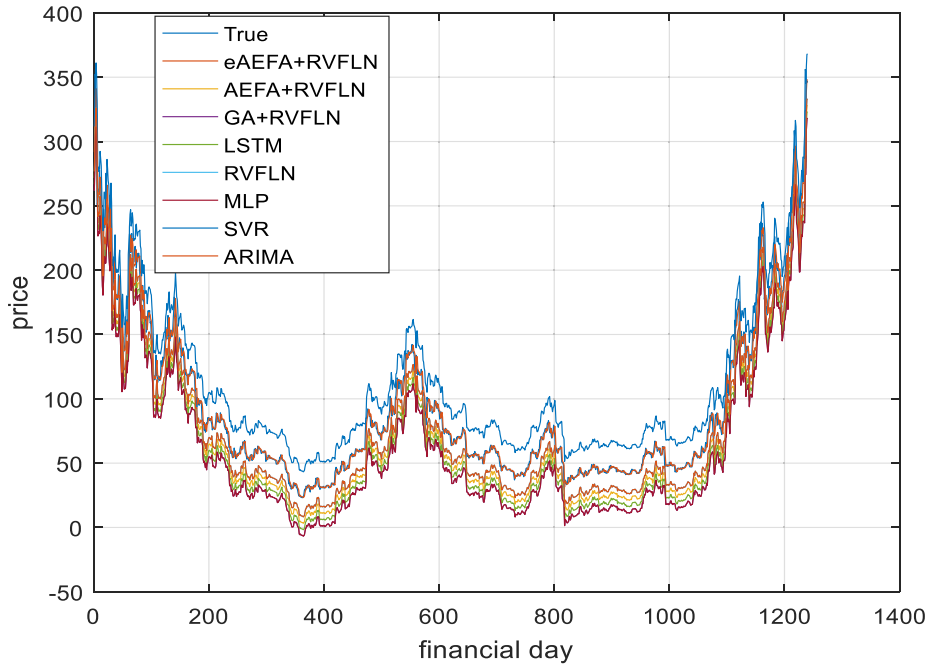

FIGURE 16. Actual v/s estimated from LTC forecasting.

increased by a lower error value. The closing prices of six series are predicted for one day in advance.

A. PERFORMANCE METRICS

The five performance measures indicated in Table 8 such as POCID, R^2, MAPE, RMSE, and MAE are used to evaluate the performance of all models.

B. EXPERIMENTAL DESIGN

The parameter values of different forecasts are chosen carefully without compromising their accuracy. We note that, all the models implemented are stochastic in nature. The optimal parameter values for all the models are selected experimentally (i.e., trial and error method) during the model training as there is no standard rule to select the optimal values of

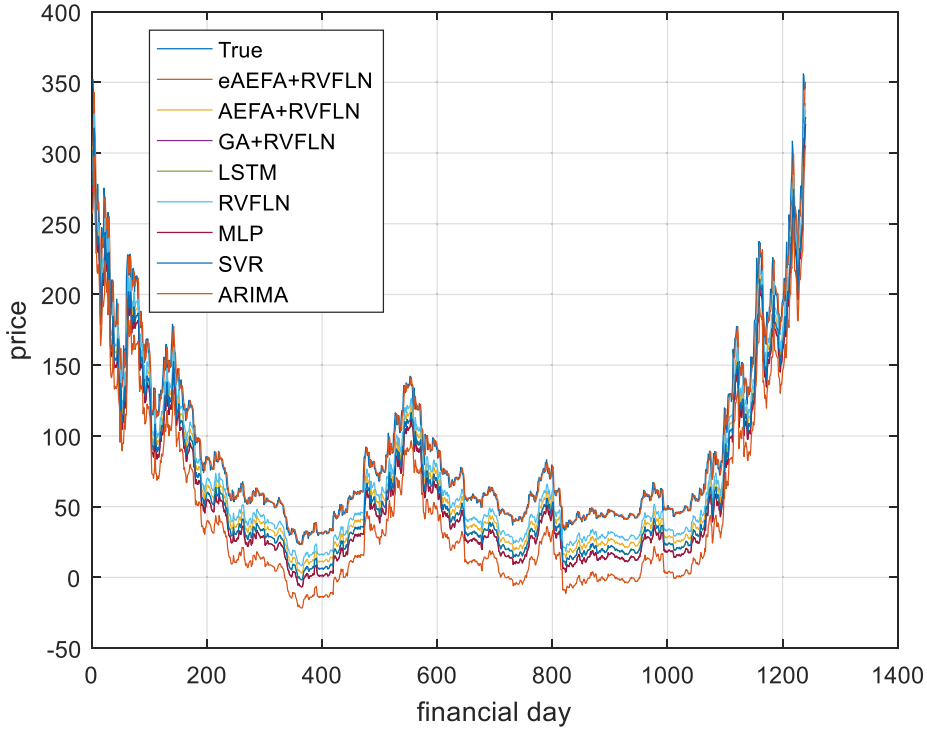

FIGURE 17. Actual v/s estimated from ETH forecasting.

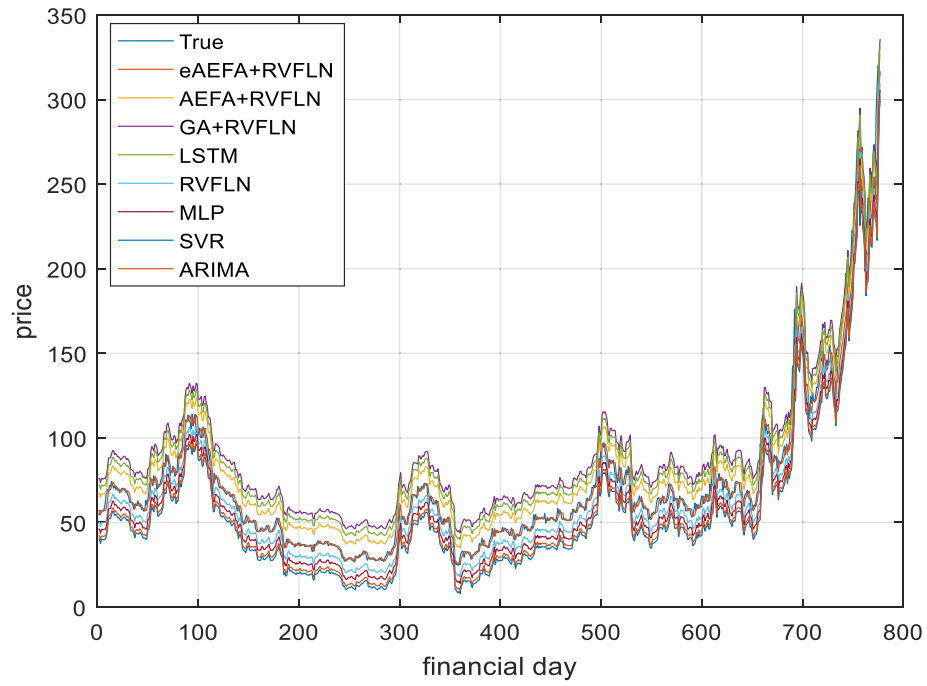

FIGURE 18. Actual v/s estimated from ZEC forecasting.

parameters for a stochastic model. Rigorous experimentations are carried out to select the optimal values of the parameters during the model training. Once the optimal values are selected at the end of training process, they are kept fixed and used for the test process. For ARIMA model, we considered $p = 1$, $d = 1$, *and* $q = 2$ (single autoregressive term, one nonseasonal difference term and two lagged forecast error). The MLP is implemented using a single hidden layer, and its size is determined empirically. With 20 to 50 neurons investigated, the ideal design is determined to be 5-32-1. (i.e., 5 input

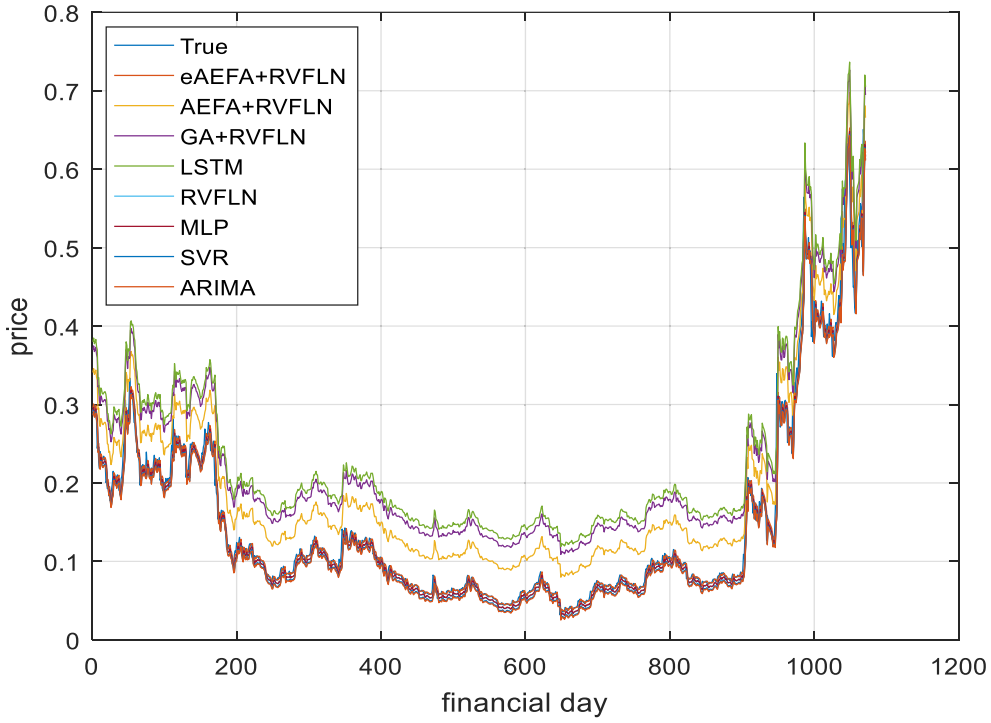

FIGURE 19. Actual v/s estimated from XLM forecasting.

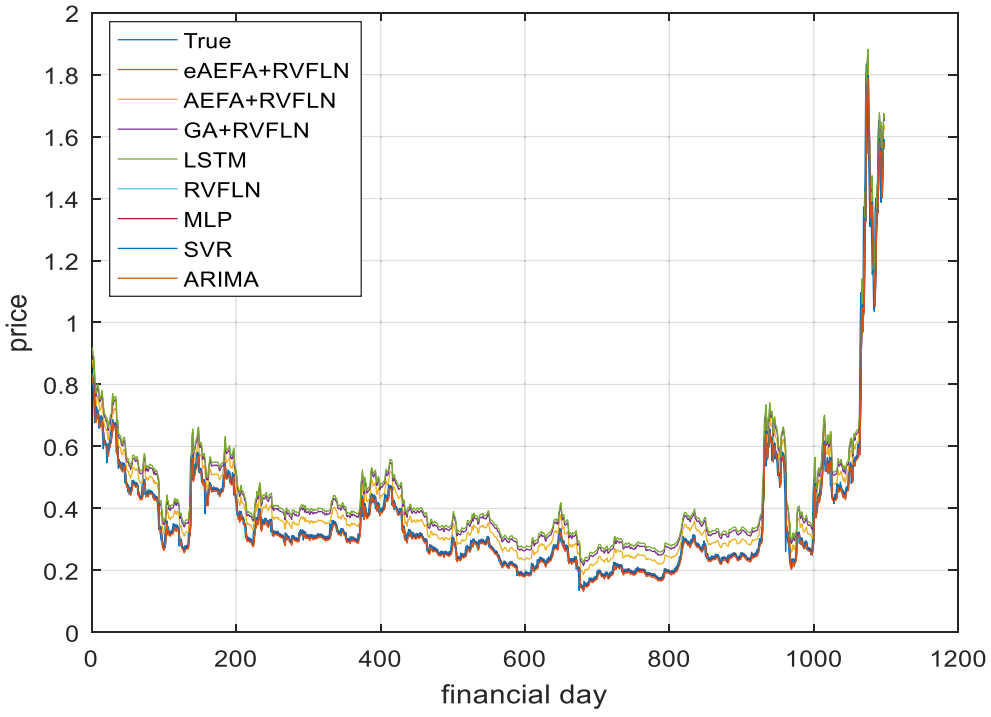

FIGURE 20. Actual v/s estimated from XRP forecasting.

layer neurons, 32 hidden layer neurons and 1 output neuron). It is trained by back propagation learning with learning rate $\alpha = 0.03$ and momentum factor of $\mu = 0.001$. For MLP and RVFLN, the number of input neuron is equals to the input data size and there is a single neuron at output layer. The enhancement layer size of RVFLN is explored similar to MLP

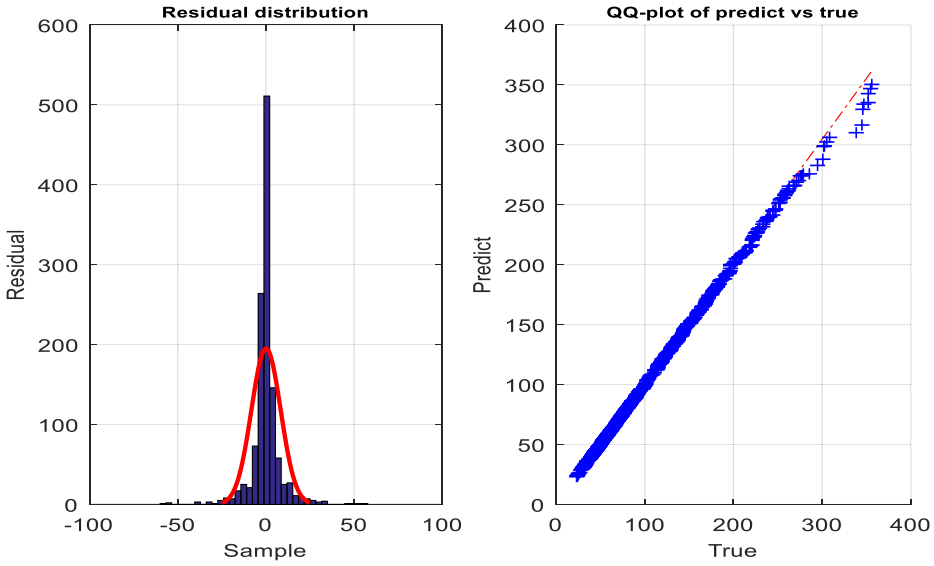

FIGURE 21. Error distributions and eAEFA+RVFLN predictions vs true closing prices from BTC series.

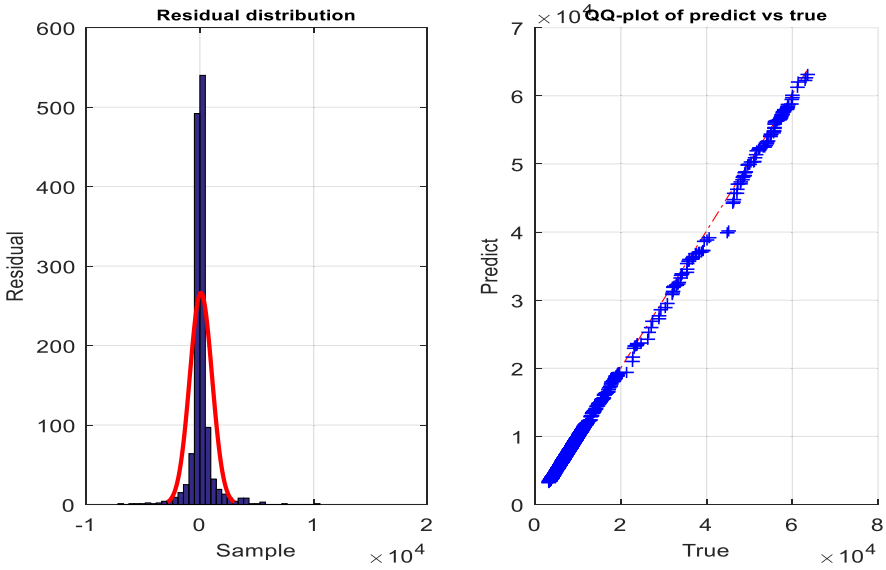

FIGURE 22. Error distributions and eAEFA+RVFLN predictions vs true closing prices from LTC series.

and set to 20. The AEFA parameters were set as particle size 50, $\alpha = 30$ and $K_0 = 500$ referring base articles [57]. The SVR adopted used polynomial kernel of degree 5, gamma = auto, tolerance = 0.001, and regularization parameter $C = 100$. For LSTM, the size of convolutional layer, *lstm* layer, and dense layer were set to 64, 72, and 16 respectively. Both convolutional and dense layer used *ReLU* activations. The learning rate is chosen as 0.0001. The elitism factor was set to 5% and the algorithm was iterated fifty times to reach the optimal parameter values. For PSO, the inertia weight, cognitive and social parameter values are set to 0.736248, 1.42374, and 1.42374 respectively. For GA, the crossover and mutation probability values were set to 0.7 and 0.003 respectively and it was iterated 100 generations. To compensate the stochastic behavior of neural-based forecasts, each one is executed 30 times with the above-mentioned parameters and random initial weight and threshold values. The average prediction error values from 30 runs are recorded for performance comparisons.

C. EXPERIMENTAL RESULTS

Tables 9 – 13 summarizes error statistics from forecasts made over six datasets. For each dataset, we perform a separate ranking of models. With 1 being the best and 8 being the

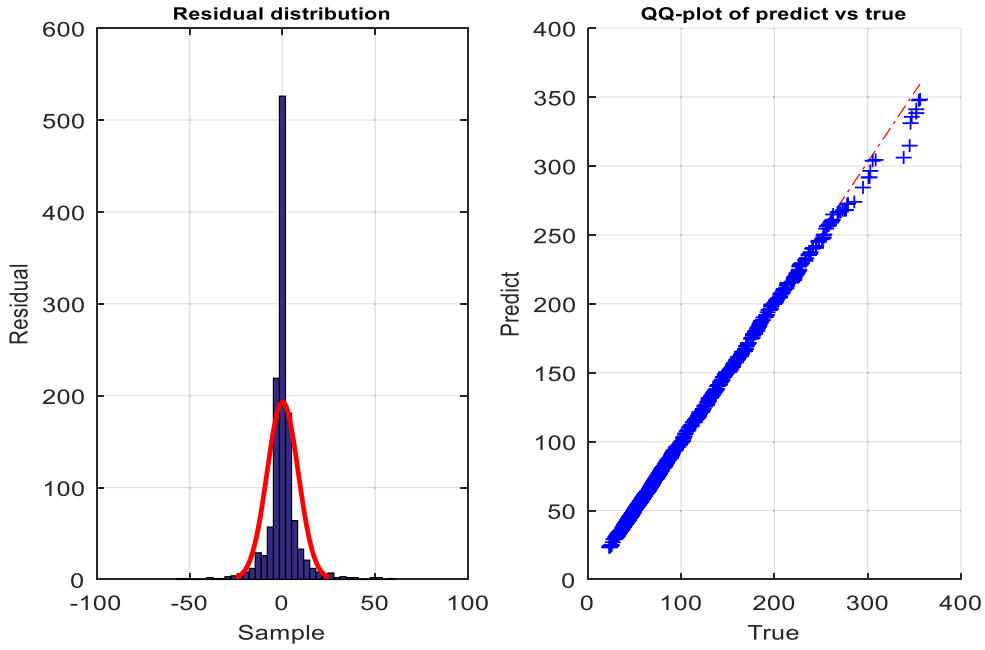

FIGURE 23. Error distributions and eAEFA+RVFLN predictions vs true closing prices from ETH series.

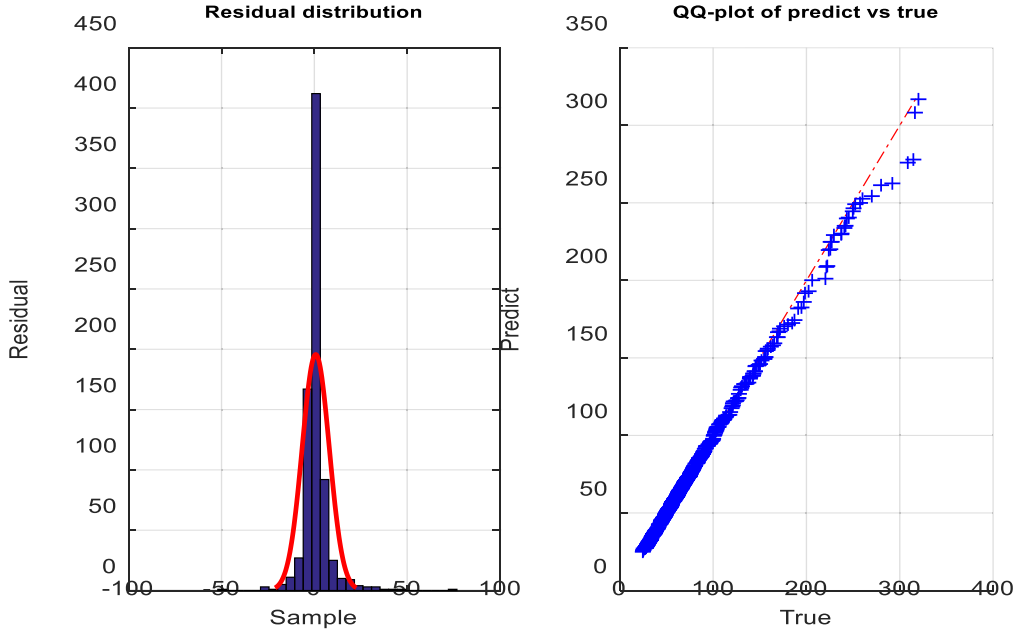

FIGURE 24. Error distributions and eAEFA+RVFLN predictions vs true closing prices from ZEC series.

worst, the rating is done from 1 to 8. An average value is supplied if the ranks are the same. Each model is given a ranking here depending on the value of its error statistic. The average rank and re-ranking of all models are displayed in the final two columns of Table 9 - 13. The reranked matrix for the employed models and metrics is shown in Table 13. Let's say that r_i^j signifies rank and that j for the chosen method falls between $(1,\ k)$ and i between $(1,\ N)$ for the chosen dataset. Here, $R_j = \frac{1}{N}\sum_i r_i^j$ stands for an algorithm's mean rank.

D. DISCUSSION

Considering the results summarized in Table 9, it is observed that the proposed forecast generated the lowest MAPE values compared to others. Its performance on XLM series is exceptional where, AEFA+RVFLN is found better. There is

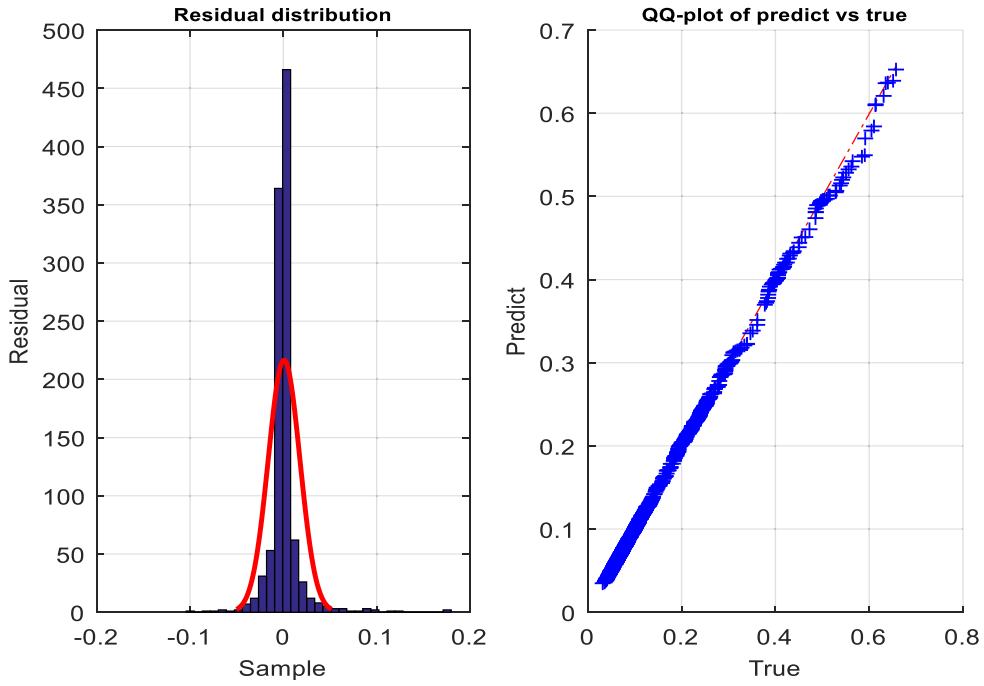

FIGURE 25. Error distributions and eAEFA+RVFLN predictions vs true closing prices from XLM series.

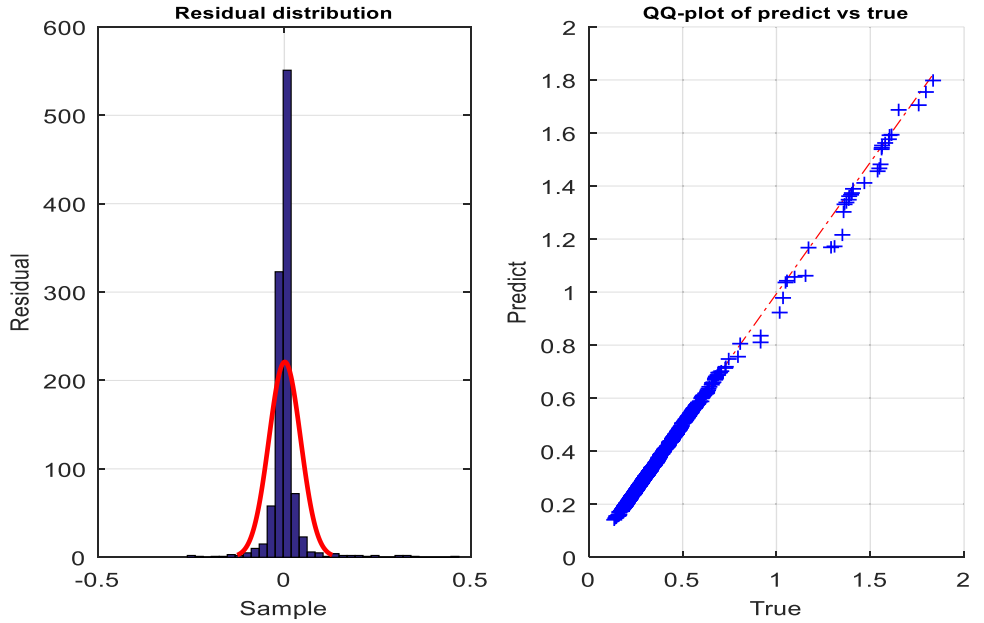

FIGURE 26. Error distributions and eAEFA+RVFLN predictions vs true closing prices from XRP series.

no significant difference in MAPE values of AEFA+RVFLN and GA+RVFLN. Both ARIMA and SVR showed inferior performances to neural network-based models. LSTM yielded acceptable MAPE values however, inferior performance compared to hybrid models. It concluded that hybrid neural models could generate more accurate predictions. For example, eAEFA+RVFLN generated a MAPE value of 0.0260 for LTC followed by GA+RVFLN with MAPE value of 0.0558, AEFA+RVFLN with MAPE value of 0.0585, LSTM and RVFLN with MAPE value of 0.0762 each, SVR with 0.0793, MLP with 0.1827, and ARIMA with 0.4427. The re-ranking value of eAEFA+RVFLN is 1 (one) followed by AEFA+RVFLN and GA+RVFLN.

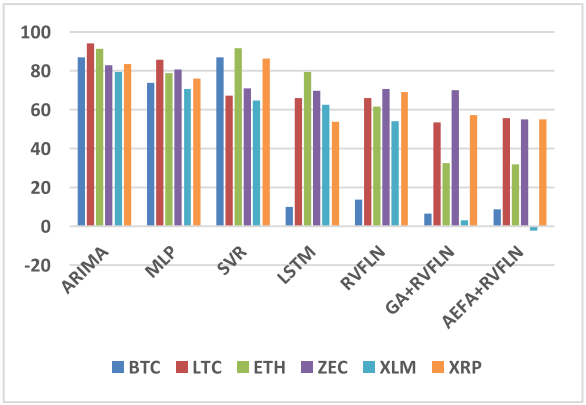

FIGURE 27. MAPE reduction (%) on adopting eAEFA+RVFLN.

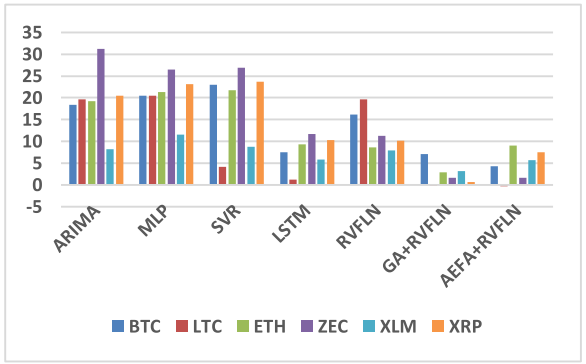

FIGURE 28. R^2 reduction (%) on adopting eAEFA+RVFLN.

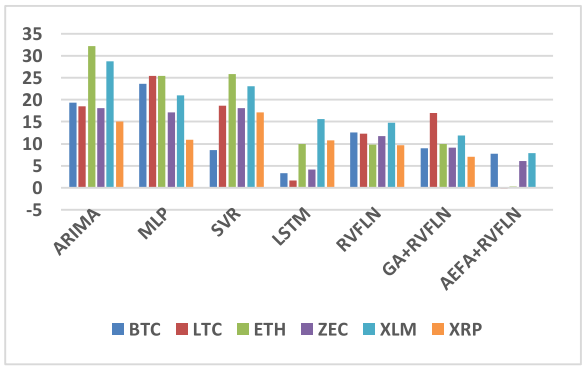

FIGURE 29. POCID reduction (%) on adopting eAEFA+RVFLN.

From R^2 metrics shown in Table 10, similar trend has seen. Here, eAEFA+RVFLN stood first again, followed by GA+RVFLN and AEFA+RVFLN. MLP and ARIMA performances are close to each other. In Table 11 and Table 13, again eAEFA+RVFLN performed best. However, in Table 12 (RMSE) it secured second rank. Here, AEFA+RVFLN obtained best overall rank. Similar findings are obtained from other tables. According to Table 14, eAEFA+RVFLN-based forecasting achieved rank one. Hence, the eAEFA+RVFLN model performs better overall than others.

Figure 15 – 20 show plots of anticipated prices versus actual prices to demonstrate the effectiveness of the suggested forecasts. These charts demonstrate how well the

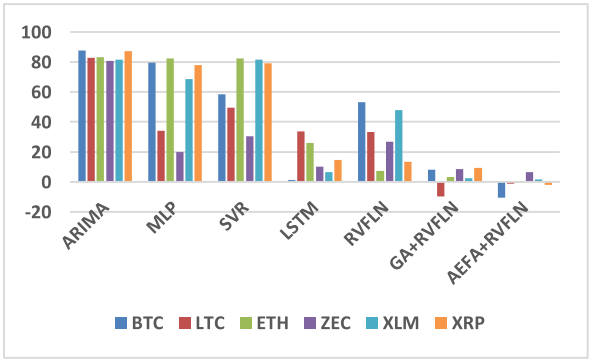

FIGURE 30. RMSE reduction (%) on adopting eAEFA+RVFLN.

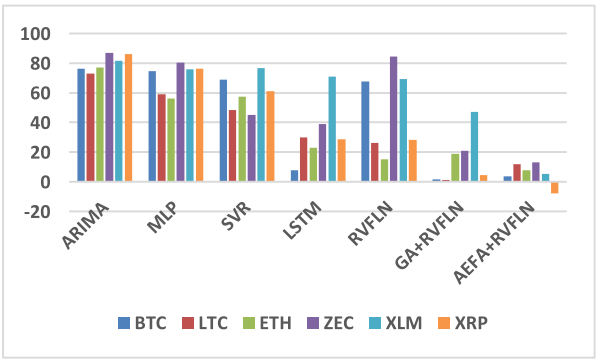

FIGURE 31. MAE reduction (%) on adopting eAEFA+RVFLN.

eAEFA+RVFLN estimates match the actuals. According to Figure 21 - 26, which display the error distribution graphs for six data series, the majority of training patterns tend to provide prediction errors that are close to zero. The reduction in error metrics on adopting eAEFA+RVFLN over others calculated by Eq. 38 are illustrated in Figure 27 – 31. It can be seen that except nominal cases, the reduction in error metrics is significant.

$$\begin{aligned} &\textit{error metric reduction} \\ &= \frac{|\textit{error of comparative model} - \textit{error of proposed model}|}{\textit{error of comparative model}} \\ &\times 100 \end{aligned} \tag{38}$$

VI. STATISTICAL SIGNIFICANCE TEST OF FORECASTS

To determine whether there exists statistically significant difference among performances of forecasts, we conducted Friedman's test which is a non-parametric test. The null and alternative hypothesis are as follows.

H0: all forecasts are equivalent and their ranks are equal.

Halt: performances of forecasts are significantly different.

In our study, for number of forecast $k = 8$, number of performance metric $N = 5$, and considering the model ranks from Table 8, the Friedman statistic $\chi_F^2 = \left[\frac{12}{[N*k*(k+1)]}\right] * \sum R^2 - [3*N*(k+1)] = \left[\frac{12}{[5*8*9)]}\right] * R^2 - [3*5*(8+1)] = 2.1156$. And the F-distribution $F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2} = \frac{(5-1)*2.1156}{5*(8-1)-2.1156} = 0.2573$.

TABLE 15. Statistics from Wilcoxon signed test for eAEFA+RVFLN.

Analyzed methods		*p*-value					
		BTC	LTC	ETH	ZEC	XML	XRP
eAEFA+RVFLN	**ARIMA**	3.25e-5 (h = 1)	3.12e-4 (h = 1)	3.24e-3 (h = 1)	3.23e-5 (h = 1)	1.20e-4 (h = 1)	3.25e-5 (h = 1)
	MLP	3.13e-3 (h = 1)	3.28e-5 (h = 1)	3.31e-3 (h = 1)	3.25e-3 (h = 1)	2.52e-2 (h = 1)	2.13e-3 (h = 1)
	SVR	3.32e-3 (h = 1)	0.2357 (h = 1)	3.12e-2 (h = 1)	3.25e-2 (h = 1)	2.15e-3 (h = 1)	3.56e-3 (h = 1)
	LSTM	3.44e-3 (h = 1)	3.62e-2 (h = 1)	2.55e-3 (h = 1)	3.04e-3 (h = 1)	2.75e-4 (h = 1)	2.86e-3 (h = 1)
	RVFLN	3.34e-4 (h = 1)	2.35e-3 (h = 1)	2.58e-2 (h = 1)	2.82 (h = 1)	1.75e-1 (h = 1)	2.34e-4 (h = 1)
	GA+RVFLN	3.52e-3 (h = 1)	0.0035 (h = 0)	3.72e-2 (h = 1)	4.20e-2 (h = 1)	3.25e-3 (h = 1)	2.52e-3 (h = 1)
	AEFA+RVFLN	2.63e-4 (h = 1)	3.25e-2 (h = 1)	2.76e-5 (h = 1)	3.30e-3 (h = 1)	1.72e-5 (h = 1)	4.02e-4 (h = 1)

Here, $2.1156 \gg 0.2573$, i.e., $\chi_F^2 > F_F$ therefore, null hypothesis of equivalence is rejected.

Since the cryptocurrency datasets used are not normally distributed, we conducted another nonparametric post hoc test, i.e., Nemenyi post hoc test to ascertain whether there is a statistical difference between two forecasts. When the grade difference between the worst-performing model and the model under examination is more than the crucial difference (CD) value, the null hypothesis of no difference is rejected. In our study, we took into account N performance measures and k number of models, Table B.16 of [60] and $\alpha = 0.01$, $q_\alpha$is found to be 0.4643 and the corresponding $CD = q_\alpha\sqrt{\frac{k(k+1)}{6N}} = 0.4643 * \sqrt{\frac{8*(8+1)}{6*5}} = 0.7192$.

Considering ARIMA as the worst performing model with average rank of 7.482, the significant difference between of models are computed as follows. Except SVR, all other models satisfy the post hoc test among which the eAEFA+RVFLN found best performing model.

$$\begin{aligned}
&AverageRank(ARIMA) - AverageRank(MLP)\\
&= 7.4826.464 = 1.018 > CD.\\
&AverageRank\ (ARIMA) - AverageRank\ (SVR)\\
&= 7.4826.898 = 0.584 < CD.\\
&AverageRank(ARIMA) - AverageRank(LSTM)\\
&= 7.4824.28 = 3.202 > CD.\\
&AverageRank(ARIMA) - AverageRank(RVFLN)\\
&= 7.4824.462 = 3.02 > CD.AverageRank(ARIMA)\\
&\quad - AverageRank(GA + RVFLN) = 7.4822.948\\
&= 4.534 > CD.\\
&AverageRank(ARIMA) - AverageRank(AEFA + RVFLN)\\
&= 7.4822.096 = 5.386 > CD.\\
&AverageRank(ARIMA) - AverageRank(eAEFA + RVFLN)\\
&= 7.4821.346 = 6.136 > CD.
\end{aligned}$$

Next, another significance test using the Wilcoxon signed-rank method is performed. The disparity between the proposed and feasible models can be attributed to a zero medians distribution, according to the results of the paired, two-sided test for the null hypothesis. Logic value $h = 1$ indicates rejection of the null hypothesis. Table 15 provides an overview of the Wilcoxon signed-rank test outcomes for the eAEFA+RVFLN dataset. These statistical results demonstrate how significantly different from other models the suggested model is.

VII. CONCLUSION

In this article, a hybrid forecast known as eAEFA+RVFLN is suggested. First, an upgraded variant known as eAEFA is aimed to enhance the performance of AEFA by combining it with the elitism mechanism. Second, the eAEFA is used to train a flat and computationally effective RVFLN, which computes the output layer weights non-iteratively using least square methods and assigns input layer parameters at random without additional modification. The next goal of this study was to evaluate how well the eAEFA+RVFLN forecast performed in terms of forecasting six cryptocurrencies. Using the identical input patterns, seven comparable models, including MLP, SVR, ARIMA, LSTM, basic RVFLN, GA+RVFLN, and AEFA+RVFLN are built and assessed. The models are assessed through five performance measures. The outcomes from exhaustive simulations and statistical significance test results have proven that eAEFA+RVFLN based forecasting superiority exists when taking into account all six cryptocurrency data series. Thus, it is determined that the eAEFA+RVFLN is best performing model and suitable for capturing the unpredictability of cryptocurrency data. Overall, adaptation of elitism concept and eAEFA based RVFLN training empowered the resulted eAEFA+RVFLN based hybrid forecast robust and competent in modelling the dynamic cryptocurrency data. Considering six datasets, it achieved an average MAPE of 0.0573, R^2 of 0.9589, POCID of 0.9676, RMSE of 0.0685, and MAE of 0.0727. Amongst the eight forecasting models developed, it secured

an average rank of 1.346 which is the best. Our models are designed to predict daily prices only. Prediction of hourly prices might be helpful to the investors. Also, along with the historical prices, several other factors affecting crypto data such as social media trend, regulations, investor sentiments and interdependent relationships among available currencies could be integrated for model training that might produce more accurate predictions. The proposed model is selecting optimal number of enhancement nodes empirically. As the size of enhancement layer plays a vital role on RVFLN performance, its selection process could be automated. Further performance enhancement on AEFA can be achieved with oppositional based learning concept. Variants of RVFLN, new modifications, and its applicability to different domains are the further interest. The current study can be extended with forecasting few other cryptocurrencies to ascertain the predictability of the proposed model.

REFERENCES

[1] J. Lansky, "Possible state approaches to cryptocurrencies," *J. Syst. Integr.*, vol. 9, no. 1, pp. 19–31, Jan. 2018.

[2] S. Nakamoto, "Re: Bitcoin P2P e-cash paper," in *The Cryptography Mailing List*, 2008, pp. 1–2.

[3] N. A. Kyriazis, "A survey on efficiency and profitable trading opportunities in cryptocurrency markets," *J. Risk Financial Manage.*, vol. 12, no. 2, p. 67, Apr. 2019.

[4] U. Mukhopadhyay, A. Skjellum, O. Hambolu, J. Oakley, L. Yu, and R. Brooks, "A brief survey of cryptocurrency systems," in *Proc. 14th Annu. Conf. Privacy, Secur. Trust (PST)*, Dec. 2016, pp. 745–752.

[5] M. Ferreira, S. Rodrigues, C. Reis, and M. Maximiano, "Blockchain: A tale of two applications," *Appl. Sci.*, vol. 8, no. 9, p. 1506, Sep. 2018.

[6] R. Grinberg, "Bitcoin: An innovative alternative digital currency," *Hastings Sci. Tech. LJ*, vol. 4, p. 159, Jan. 2012.

[7] F. Mai, Z. Shan, Q. Bai, X. Wang, and R. H. L. Chiang, "How does social media impact Bitcoin value? A test of the silent majority hypothesis," *J. Manage. Inf. Syst.*, vol. 35, no. 1, pp. 19–52, Jan. 2018.

[8] A. Corelli, "Cryptocurrencies and exchange rates: A relationship and causality analysis," *Risks*, vol. 6, no. 4, p. 111, Oct. 2018.

[9] N. Trabelsi, "Are there any volatility spill-over effects among cryptocurrencies and widely traded asset classes?" *J. Risk Financial Manage.*, vol. 11, no. 4, p. 66, Oct. 2018.

[10] N. Gradojevic, D. Kukolj, R. Adcock, and V. Djakovic, "Forecasting Bitcoin with technical analysis: A not-so-random forest?" *Int. J. Forecasting*, vol. 39, no. 1, pp. 1–17, Jan. 2023.

[11] N. Rathee, A. Singh, T. Sharda, N. Goel, M. Aggarwal, and S. Dudeja, "Analysis and price prediction of cryptocurrencies for historical and live data using ensemble-based neural networks," *Knowl. Inf. Syst.*, May 2023, doi: 10.1007/s10115-023-01871-0.

[12] D. Shang, Z. Yan, L. Zhang, and Z. Cui, "Digital financial asset price fluctuation forecasting in digital economy era using blockchain information: A reconstructed dynamic-bound Levenberg–Marquardt neural-network approach," *Expert Syst. Appl.*, vol. 228, Oct. 2023, Art. no. 120329.

[13] M. Ortu, N. Uras, C. Conversano, S. Bartolucci, and G. Destefanis, "On technical trading and social media indicators for cryptocurrency price classification through deep learning," *Expert Syst. Appl.*, vol. 198, Jul. 2022, Art. no. 116804.

[14] K. Murray, A. Rossi, D. Carraro, and A. Visentin, "On forecasting cryptocurrency prices: A comparison of machine learning, deep learning, and ensembles," *Forecasting*, vol. 5, no. 1, pp. 196–209, Jan. 2023.

[15] D. Selvamuthu, V. Kumar, and A. Mishra, "Indian stock market prediction using artificial neural networks on tick data," *Financial Innov.*, vol. 5, no. 1, pp. 1–12, Dec. 2019.

[16] A. H. Moghaddam, M. H. Moghaddam, and M. Esfandyari, "Stock market index prediction using artificial neural network," *J. Econ., Finance Administ. Sci.*, vol. 21, no. 41, pp. 89–93, Dec. 2016.

[17] A. Misnik, S. Krutalevich, S. Prakapenka, P. Borovykh, and M. Vasiliev, "Impact analysis of additional input parameters on neural network cryptocurrency price prediction," in *Proc. 21st Int. Conf. Complex Syst., Control Modeling Problems (CSCMP)*, Sep. 2019, pp. 163–167.

[18] P. Jay, V. Kalariya, P. Parmar, S. Tanwar, N. Kumar, and M. Alazab, "Stochastic neural networks for cryptocurrency price prediction," *IEEE Access*, vol. 8, pp. 82804–82818, 2020.

[19] E. Akyildirim, A. Goncu, and A. Sensoy, "Prediction of cryptocurrency returns using machine learning," *Ann. Operations Res.*, vol. 297, nos. 1–2, pp. 3–36, Feb. 2021.

[20] A. Yazdinejad, H. HaddadPajouh, A. Dehghantanha, R. M. Parizi, G. Srivastava, and M.-Y. Chen, "Cryptocurrency malware hunting: A deep recurrent neural network approach," *Appl. Soft Comput.*, vol. 96, Nov. 2020, Art. no. 106630.

[21] L. Alessandretti, A. ElBahrawy, L. M. Aiello, and A. Baronchelli, "Anticipating cryptocurrency prices using machine learning," *Complexity*, vol. 2018, pp. 1–16, Nov. 2018.

[22] M. Nakano, A. Takahashi, and S. Takahashi, "Bitcoin technical trading with artificial neural network," *Phys. A, Stat. Mech. Appl.*, vol. 510, pp. 587–609, Nov. 2018.

[23] J. Rebane, I. Karlsson, P. Papapetrou, and S. Denic, "Seq2Seq RNNs and ARIMA models for cryptocurrency prediction: A comparative study," in *Proc. SIGKDD Fintech*, London, U.K.: Fintech, Aug. 2018.

[24] Z. Shahbazi and Y. Byun, "Improving the cryptocurrency price prediction performance based on reinforcement learning," *IEEE Access*, vol. 9, pp. 162651–162659, 2021.

[25] X. Du, Z. Tang, J. Wu, K. Chen, and Y. Cai, "A new hybrid cryptocurrency returns forecasting method based on multiscale decomposition and an optimized extreme learning machine using the sparrow search algorithm," *IEEE Access*, vol. 10, pp. 60397–60411, 2022.

[26] H. Guo, D. Zhang, S. Liu, L. Wang, and Y. Ding, "Bitcoin price forecasting: A perspective of underlying blockchain transactions," *Decis. Support Syst.*, vol. 151, Dec. 2021, Art. no. 113650.

[27] R. Chowdhury, M. A. Rahman, M. S. Rahman, and M. R. C. Mahdy, "An approach to predict and forecast the price of constituents and index of cryptocurrency using machine learning," *Phys. A, Stat. Mech. Appl.*, vol. 551, Aug. 2020, Art. no. 124569.

[28] B. Kapar and J. Olmo, "Analysis of Bitcoin prices using market and sentiment variables," *World Economy*, vol. 44, no. 1, pp. 45–63, Jan. 2021.

[29] Z. H. Munim, M. H. Shakil, and I. Alon, "Next-day Bitcoin price forecast," *J. Risk Financial Manage.*, vol. 12, no. 2, p. 103, Jun. 2019.

[30] N. Uras, L. Marchesi, M. Marchesi, and R. Tonelli, "Forecasting Bitcoin closing price series using linear regression and neural networks models," *PeerJ Comput. Sci.*, vol. 6, p. e279, Jul. 2020.

[31] L. Cocco, R. Tonelli, and M. Marchesi, "Predictions of Bitcoin prices through machine learning based frameworks," *PeerJ Comput. Sci.*, vol. 7, p. e413, Mar. 2021.

[32] D.-T. Nguyen and H.-V. Le, "Predicting the price of Bitcoin using hybrid ARIMA and machine learning," in *Proc. Int. Conf. Future Data Secur. Eng.* Cham, Switzerland: Springer, 2019, pp. 696–704.

[33] R. Bohte and L. Rossini, "Comparing the forecasting of cryptocurrencies by Bayesian time-varying volatility models," *J. Risk Financial Manage.*, vol. 12, no. 3, p. 150, Sep. 2019.

[34] J. Yoon, "Forecasting of real GDP growth using machine learning models: Gradient boosting and random forest approach," *Comput. Econ.*, vol. 57, no. 1, pp. 247–265, Jan. 2021.

[35] O. Sohaib, W. Hussain, M. Asif, M. Ahmad, and M. Mazzara, "A PLS-SEM neural network approach for understanding cryptocurrency adoption," *IEEE Access*, vol. 8, pp. 13138–13150, 2020.

[36] S. Alonso-Monsalve, A. L. Suárez-Cetrulo, A. Cervantes, and D. Quintana, "Convolution on neural networks for high-frequency trend prediction of cryptocurrency exchange rates using technical indicators," *Expert Syst. Appl.*, vol. 149, Jul. 2020, Art. no. 113250.

[37] S. C. Nayak, S. Dehuri, and S. Cho, "Intelligent financial forecasting with an improved chemical reaction optimization algorithm based dendritic neuron model," *IEEE Access*, vol. 10, pp. 130921–130943, 2022.

[38] S. C. Nayak, "Bitcoin closing price movement prediction with optimal functional link neural networks," *Evol. Intell.*, vol. 15, no. 3, pp. 1825–1839, Sep. 2022.

[39] S. Das, S. C. Nayak, and B. Sahoo, "Towards crafting optimal functional link artificial neural networks with RAO algorithms for stock closing prices prediction," *Comput. Econ.*, vol. 60, no. 1, pp. 1–23, Jun. 2022.

[40] Y.-H. Pao, G.-H. Park, and D. J. Sobajic, "Learning and generalization characteristics of the random vector functional-link net," *Neurocomputing*, vol. 6, no. 2, pp. 163–180, Apr. 1994.

[41] I. Majumder, P. K. Dash, and R. Bisoi, "Short-term solar power prediction using multi-kernel-based random vector functional link with water cycle algorithm-based parameter optimization," *Neural Comput. Appl.*, vol. 32, no. 12, pp. 8011–8029, Jun. 2020.

[42] R. Katuwal and P. N. Suganthan, "Stacked autoencoder based deep random vector functional link neural network for classification," *Appl. Soft Comput.*, vol. 85, Dec. 2019, Art. no. 105854.
[43] R. Bisoi, P. K. Dash, and S. P. Mishra, "Modes decomposition method in fusion with robust random vector functional link network for crude oil price forecasting," *Appl. Soft Comput.*, vol. 80, pp. 475–493, Jul. 2019.
[44] Y. Ren, P. N. Suganthan, N. Srikanth, and G. Amaratunga, "Random vector functional link network for short-term electricity load demand forecasting," *Inf. Sci.*, vols. 367–368, pp. 1078–1093, Nov. 2016.
[45] P. Dai, F. Gwadry-Sridhar, M. Bauer, M. Borrie, and X. Teng, "Healthy cognitive aging: A hybrid random vector functional-link model for the analysis of Alzheimer's disease," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 4567–4573.
[46] Z. Wang, S. Yoon, S. J. Xie, Y. Lu, and D. S. Park, "A high accuracy pedestrian detection system combining a cascade AdaBoost detector and random vector functional-link net," *Sci. World J.*, vol. 2014, pp. 1–7, Jan. 2014.
[47] L. Zhang and P. N. Suganthan, "Visual tracking with convolutional random vector functional link network," *IEEE Trans. Cybern.*, vol. 47, no. 10, pp. 3243–3253, Oct. 2017.
[48] S. Scardapane, D. Comminiello, M. Scarpiniti, and A. Uncini, "A semi-supervised random vector functional-link network based on the transductive framework," *Inf. Sci.*, vols. 364–365, pp. 156–166, Oct. 2016.
[49] W. Dai, Q. Liu, and T. Chai, "Particle size estimate of grinding processes using random vector functional link networks with improved robustness," *Neurocomputing*, vol. 169, pp. 361–372, Dec. 2015.
[50] D. P. P. Mesquita, J. P. P. Gomes, L. R. Rodrigues, S. A. F. Oliveira, and R. K. H. Galvão, "Building selective ensembles of randomization based neural networks with the successive projections algorithm," *Appl. Soft Comput.*, vol. 70, pp. 1135–1145, Sep. 2018.
[51] R. Katuwal and P. N. Suganthan, "Dropout and DropConnect based ensemble of random vector functional link neural network," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Nov. 2018, pp. 1772–1778.
[52] X. Qiu, P. N. Suganthan, and G. A. J. Amaratunga, "Electricity load demand time series forecasting with empirical mode decomposition based random vector functional link network," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2016, pp. 1394–1399.
[53] X. Qiu, P. N. Suganthan, and G. A. J. Amaratunga, "Ensemble incremental learning random vector functional link network for short-term electric load forecasting," *Knowl.-Based Syst.*, vol. 145, pp. 182–196, Apr. 2018.
[54] X.-S. Yang, "Nature-inspired optimization algorithms: Challenges and open problems," *J. Comput. Sci.*, vol. 46, Oct. 2020, Art. no. 101104.
[55] A. Darwish, "Bio-inspired computing: Algorithms review, deep analysis, and the scope of applications," *Future Comput. Informat. J.*, vol. 3, no. 2, pp. 231–246, Dec. 2018.
[56] S. C. Nayak and B. B. Misra, "A chemical-reaction-optimization-based neuro-fuzzy hybrid network for stock closing price prediction," *Financial Innov.*, vol. 5, no. 1, pp. 1–34, Dec. 2019.
[57] A. Yadav, "AEFA: Artificial electric field algorithm for global optimization," *Swarm Evol. Comput.*, vol. 48, pp. 93–108, Aug. 2019.
[58] R. V. Rao and V. Patel, "An elitist teaching-learning-based optimization algorithm for solving complex constrained optimization problems," *Int. J. Ind. Eng. Comput.*, vol. 3, no. 4, pp. 535–560, Jul. 2012.

SARAT CHANDRA NAYAK received the M.Tech. degree in computer science from Utkal University, Bhubaneswar, India, in 2011, and the Ph.D. degree in computer engineering from the Veer Surendra Sai University of Technology (VSSUT), Burla, India, in 2016. He has 15 years of experience in teaching and research. Currently, he is associated with the Soft Computing Laboratory, Department of Computer Science, Yonsei University, South Korea, as a Postdoctoral Research Fellow with the Brain Korea 21 (BK21) Fellowship. He has more than 80 research articles in reputed international journals and conferences, one book, and 14 book chapters in his credit. His research interests include machine learning, data mining, soft computing, predictive systems, financial time series forecasting, computational intelligence, and evolutionary computations.

SUBHRANGINEE DAS received the M.Tech. degree in computer science from Utkal University, Bhubaneswar, India, in 2009, and the Ph.D. degree in computer science and engineering from KIIT University, Bhubaneswar, in 2022. She is an Assistant Professor with the Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Hyderabad. She was a Visiting Faculty with Utkal University, from 2013 to 2014. She has more than eight research articles in reputed international journals and conferences. Her research interests include computational intelligence, data mining, soft computing, and financial time series forecasting.

SATCHIDANANDA DEHURI (Senior Member, IEEE) received the Ph.D. degree in computer science from Utkal University, Vani Vihar, Odisha, in 2006. He has been a Professor with the Department of Computer Science, Fakir Mohan University, Balasore, Odisha, India, since 2013. He visited the Soft Computing Laboratory, Yonsei University, Seoul, South Korea, as a BOYSCAST Fellow, under the BOYSCAST Fellowship Program of DST, Government of India, in 2008. His research interests include multi-objective optimization, machine learning, and data science. In 2010, he received the Young Scientist Award in Engineering and Technology for the year 2008 from the Department of Science and Technology, Odisha Vigyan Academy, Government of Odisha. In 2021, he received the Teachers Associate ship and Research Excellence (TARE) Fellowship from SERB, DST, Government of India, for three years to carry out intensive research on higher order neural networks for big data analysis at host Institute, ISI Kolkata and Parent Institute, Fakir Mohan University, Balasore.

SUNG-BAE CHO (Senior Member, IEEE) received the B.S. degree in computer science from Yonsei University, Seoul, South Korea, and the M.S. and Ph.D. degrees in computer science from the Korea Advanced Institute of Science and Technology (KAIST), Taejeon, South Korea. He was an Invited Researcher with the Human Information Processing Research Laboratories, Advanced Telecommunications Research (ATR) Institute, Kyoto, Japan, from 1993 to 1995; and a Visiting Scholar with the University of New South Wales, Canberra, Australia, in 1998. He was also a Visiting Professor with the University of British Columbia, Vancouver, Canada, from 2005 to 2006; and King Mongkut's University of Technology, Thonburi, Bangkok, Thailand, in 2013. Since 1995, he has been a Professor with the Department of Computer Science, Yonsei University, and a Underwood Distinguished Professor, since 2021. He has published over 230 journal articles and over 680 conference papers. His research interests include neural networks, pattern recognition, intelligent man–machine interfaces, evolutionary computation, and artificial life. He was a recipient of the Richard E. Merwin Prize from the IEEE Computer Society, in 1993. He received several distinguished investigator awards from the Korea Information Science Society, in 2005, and the GaheonSindoricoh, in 2017. He was also a recipient of the Service Merit Medal from the Korean Government, in 2022.