Applied Soft Computing Journal 161 (2024) 111759

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Recurrent ensemble random vector functional link neural network for financial time series forecasting

Aryan Bhambu [a], Ruobin Gao [b], Ponnuthurai Nagaratnam Suganthan [c,*]

[a] Department of Mathematics, Indian Institute of Technology, Guwahati, 781039, Assam, India
[b] School of Civil and Environmental Engineering, Nanyang Technological University, Singapore
[c] KINDI Computing Research, College of Engineering, Qatar University, Doha, Qatar

ARTICLE INFO

Keywords:
Time series forecasting
Randomized neural network
Machine learning
Deep learning
Ensemble deep learning
Finance

ABSTRACT

Financial time series forecasting is crucial in empowering investors to make well-informed decisions, manage risks effectively, and strategically plan their investment activities. However, the non-stationary and non-linear characteristics inherent in time series data pose significant challenges when accurately predicting future forecasts. This paper proposes a novel Recurrent ensemble deep Random Vector Functional Link (RedRVFL) network for financial time series forecasting. The proposed model leverages randomly initialized and fixed weights for the recurrent hidden layers, ensuring stability during training. Furthermore, incorporating stacked hidden layers enables deep representation learning, facilitating the extraction of complex patterns from the data. The proposed model generates the forecast by combining the outputs of each layer through an ensemble approach. A comparative analysis was conducted against several state-of-the-art models over financial time-series datasets, and the results demonstrated the superior performance of our proposed model in terms of forecasting accuracy and predictive capability.

1. Introduction

Financial time series forecasting is a crucial factor in the domain of finance, offering invaluable insights and predictions essential for informed decision-making by investors, financial institutions, and analysts. The objective is to discern underlying patterns, relationships, and dynamics within the data to generate accurate forecasts vital for strategic planning, risk management, portfolio optimization, and assessing investment opportunities. However, the complexity arises from financial markets' volatile and non-linear nature, the influence of external factors, and inherent uncertainties. This specialized task within time series forecasting has seen significant advancements, utilizing sophisticated models and techniques to capture patterns and dynamics effectively. Proposed methods, categorized into statistical and artificial intelligence (AI) approaches, address the unique challenges of financial time series forecasting. Commonly employed statistical methods include Autoregressive Integrated Moving Average (ARIMA) [1], Hidden Markov Model [2], and fuzzy logic [3].

Accurate forecasting in financial time series encounters difficulties due to the complex features of robust non-linearity and non-stationarity, posing challenges for conventional statistical methods. As a result, researchers have introduced various artificial intelligence (AI) approaches tailored explicitly for financial time series forecasting. These AI-based forecasting methods encompass a range of techniques, including the Support vector regression (SVR) [4,5], Artificial Neural Networks (ANNs) [6,7], Random Vector Functional Link (RVFL) Network [8], Deep Neural Networks [9,10]. These AI methods leverage advanced computational algorithms and machine learning models to capture intricate patterns, non-linearities, and temporal dependencies within the data, thereby enhancing the forecasting accuracy for economic and financial time series.

Neural networks have gained significant popularity because of their capability to capture the hidden and non-linear pattern of time series forecasting [11–13]. Deep learning models have remarkable success across diverse fields, encompassing health [14], communication [15], and financial time series prediction [9,11,16]. The hierarchical structures of deep learning models enable them to capture meaningful representations of input data, contributing to their accurate time series forecasting capabilities. Although neural networks have dominated intelligent forecasting methods, ANNs typically lack the inherent ability to retain the memory of past inputs, which is crucial for precise time series modelling. Zhang et al. [17] compared the forecasting performance of ANNs with traditional statistical methods and found that ANNs tend to underperform in capturing complex time series patterns and long-term dependencies. Recurrent Neural Networks (RNNs) is a type of

* Corresponding author.
E-mail addresses: a.bhambu@iitg.ac.in (A. Bhambu), gaor0009@e.ntu.edu.sg (R. Gao), p.n.suganthan@qu.edu.qa (P.N. Suganthan).

https://doi.org/10.1016/j.asoc.2024.111759
Received 12 February 2024; Received in revised form 3 May 2024; Accepted 7 May 2024
Available online 16 May 2024

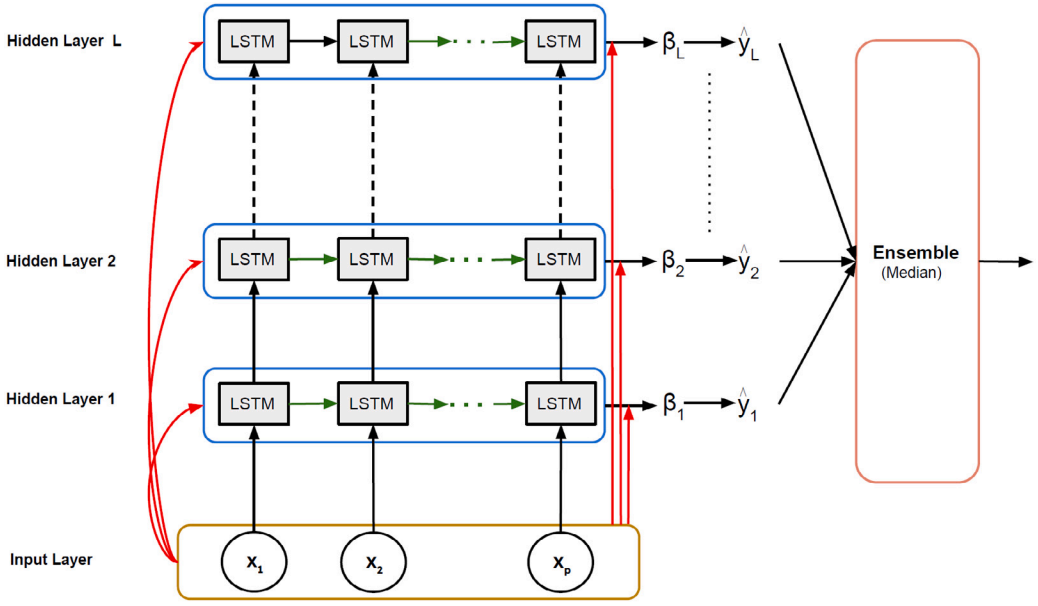

Fig. 1. Architecture of the RedRVFL having input layer and hidden layers equipped with LSTM cells.

ANN introduced for managing sequential data through the integration of feedback connections [18]. Unlike traditional feedforward ANNs, RNNs have the ability to preserve and store information from preceding steps within the sequence, allowing them to capture temporal dependencies and patterns. The critical advantage of RNNs over traditional ANNs is their ability to handle variable-length inputs and outputs, making them well-suited for time series prediction tasks [18].

However, fully trained deep learning models often encounter substantial computational burdens due to the iterative optimization process involved in weight and bias adjustments. The feedforward and back-propagation computations of output and gradients in each iteration, combined with the necessity of processing the entire dataset in multiple iterations, contribute to the computational complexity. This paper introduces an innovative and efficient ensemble deep neural network tailored to address the associated challenges for financial time series forecasting. The proposed model harnesses the strengths of both deep learning and ensemble learning while mitigating the computational load. The study explores the predictive abilities of a specific variant of randomized deep neural networks referred to as RVFL (Random Vector Functional Link) networks. The hidden layers within the RVFL network undergo random initialization and do not necessitate additional training. The randomization technique employed in RVFL networks facilitates the incorporation of strong non-linearity and generalization abilities [19]. The unique aspect necessitating training is the linear output layer, where the weights are determined through a closed-form solution, eliminating the need for iterative steps. Consequently, the training process for the deep RVFL network is significantly faster when contrasted with deep learning models because it relies on iterative training. Additionally, the deep RVFL network integrates ensemble learning techniques to enhance the robustness and reliability of predictions [20]. Combining multiple models mitigates the inherent uncertainties associated with a single model, leading to more accurate and stable forecasts.

The edRVFL network is a neural network architecture that employs randomized weights, resulting in fast training processes. However, a limitation of the edRVFL network is its inability to capture the intricate sequential patterns inherent in time series data due to the absence of feedback connections within the network. Feedback connections enable the network to incorporate the influence of previous predictions or errors, allowing for detecting and modelling complex sequential behaviours within the time series. Therefore, while the edRVFL network offers advantages in terms of training speed, it cannot effectively capture and exploit the temporal dependencies present in time series data.

In order to address the limitations associated with the edRVFL network and traditional RNNs, particularly in capturing complex temporal dependencies and high computational cost, we have proposed an ensemble deep network. Our proposed approach leverages the strengths of both edRVFL, with its ensemble learning capabilities, and Recurrent (LSTM) networks, known for their ability to model and capture long-term dependencies. Combining these two architectures aims to enhance the network's capacity to capture and preserve the complex sequential behaviour in time series data. The ensemble deep neural network enables us to harness the advantages of edRVFL and LSTMs, ultimately improving the network's performance in modelling and forecasting time series data with complex temporal dynamics.

The proposed model introduces several novel perspectives, which can be summarized as follows:

- In this paper, a novel network Recurrent ensemble deep random vector functional link (RedRVFL) is introduced, which leverages randomized and sequential networks for financial time series forecasting.
- The model employs recurrent cells to detect sequential patterns within time series data and presents an innovative ensemble architecture that integrates Recurrent connections and edRVFL for improved financial time series forecasting. Furthermore, this paper employs edRVFL for the first time in financial forecasting.
- This paper utilizes the layer-wise Bayesian Optimization technique for the hyperparameter tuning. Furthermore, this paper presents the first implementation of edRVFL for financial time series forecasting.
- The research conducts a comparative analysis with various benchmark models, encompassing both statistical methods and state-of-the-art models. This investigation is conducted across eleven diverse financial time series datasets. The assessment incorporates three error metrics and a statistical test for performance comparison. The findings from the experiments showcase the superior performance of the proposed model.

2. Related methods

This section provides a brief overview of the literature related to randomized neural networks, with a specific focus on the RVFL

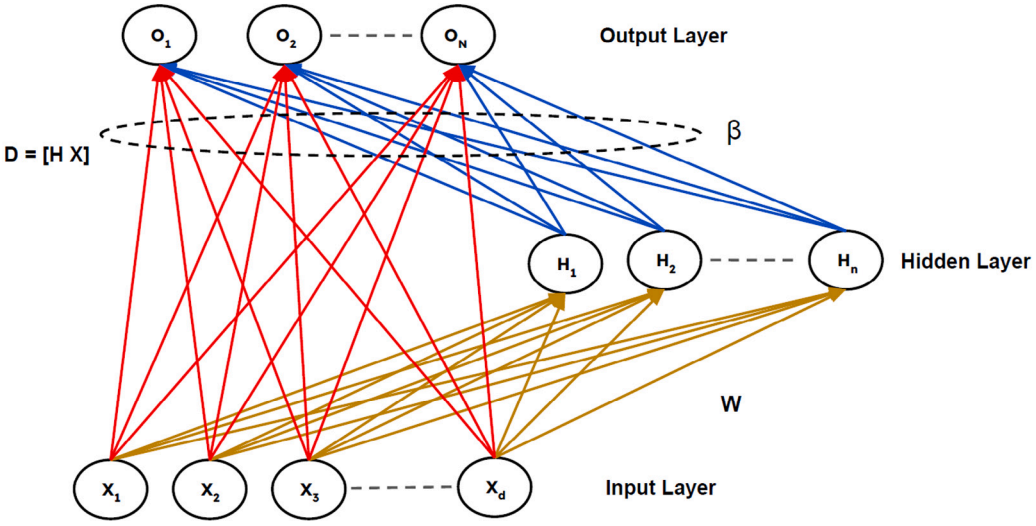

Fig. 2. Architecture of the RVFL network involves randomly initialized weights W, hidden feature H, and augmented feature D.

network, ensemble RVFL network, LSTM network, and the Bayesian Optimization method for hyperparameter optimization.

2.1. Random Vector Functional Link (RVFL)

The RVFL architecture [21] is a feedforward network consisting of a single hidden layer. Unlike conventional neural networks, the RVFL network has a specific direct link connecting the input and output layers. The hidden layer parameters of the network are initialized randomly and then remain constant. The architecture of the RVFL network, as depicted in Fig. 2, has an input, hidden, and output layer.

The non-linear hidden features generated from the hidden layer employing randomly fixed parameters can be computed using Eq. (1):

$$H = f(WX) \tag{1}$$

Here, $W \in \mathbb{R}^{d\times n}$ signifies the weight vector, while $H \in \mathbb{R}^{n\times N}$ denotes the hidden features. The augmented feature combination of input and hidden features is represented by D, and f is a non-linear activation function. Subsequently, the output weights β are determined by solving the optimization problem outlined below:

$$\mathcal{L}_{RVFL} = \min_{\beta} \|D\beta - Y\|^2 + \lambda\|\beta\|^2 \tag{2}$$

Here, Y represents the true vector to be fitted, and λ denotes the regularization parameter that governs the RVFL network's consideration of model complexity.

Typically, optimization problems of this nature can be tackled using methods such as Moore–Penrose pseudo inverse [22] and ridge regression [23]. When employing the Moore–Penrose pseudo inverse approach, the contribution of $\|\beta\|^2$ is disregarded, and λ is conventionally taken as the value zero.

$$\beta = D^{\dagger}Y \tag{3}$$

Alternatively, for ridge regression where $\lambda \neq 0$, the solution can be expressed as:

$$\text{Primal Space: } \beta = (D^T D + \lambda I)^{-1} D^T Y \tag{4}$$

$$\text{Dual Space: } \beta = D^T (DD^T + \lambda I)^{-1} Y \tag{5}$$

The choice between the primal and dual solutions depends on the total number of feature dimensions, enabling a reduction in computational complexity during RVFL training [24].

2.2. Ensemble Deep Random Vector Functional Link (edRVFL)

Deep RVFL [25], employs deep representation learning within the RVFL network. Unlike conventional RVFL, deep RVFL(dRVFL) incorporates a hierarchical structure through the stacking of multiple hidden layers. The architecture leveraging deep representation learning makes use of this hierarchical structure by sequentially stacking multiple layers. Each hidden layer produces non-linear random features by processing the input features, thereby generating a diverse set of feature representations.

Furthermore, ensemble deep RVFL is introduced, capitalizing on ensemble learning and the dRVFL network. The edRVFL network incorporates skip connections from input features and the non-linear hidden representation of each layer to generate subsequent hidden layer features. This architecture enables the network to acquire both linear and non-linear representations at each hidden layer. Unlike the dRVFL, which provides a singular set of predictions, the edRVFL generates L forecasts corresponding to each hidden layer. Unlike deep neural networks, the edRVFL model undergoes training for multiple output layers instead of a singular output layer. The final result is obtained by combining the outputs derived from each hidden layer.

The architecture of an edRVFL network, consisting of L hidden layers, each containing N hidden nodes, is illustrated in Fig. 3. Let the input feature is $X \in \mathbb{R}^{n\times d}$, then the hidden feature obtained from the initial hidden layer is computed using the equation:

$$H^1 = f(W^1 X) \tag{6}$$

Here, $W^1 \in \mathbb{R}^{d\times N}$ represents the weight vector for the initial hidden layer, d signifies the number of input features, and f denotes the non-linear activation function. The Eq. (6) can be extended to multiple layers, yielding a general formulation:

$$H^l = f(W^l[H^{(l-1)}, X]) \tag{7}$$

where $W^l \in \mathbb{R}^{(N+d)\times n}$ represents weight vector for lth hidden layer. Each hidden layer in this framework is considered as an independent regressor. For a single hidden layer, the input features from the initial layer, along with all hidden features contained within that layer, collectively serve as the input for prediction. The outputs of each hidden layer can be computed using the Eqs. (4) and (5).

2.3. Long Short-Term Memory (LSTM)

LSTM [18] is a class of RNN architecture introduced to tackle challenges such as the vanishing gradient problem, ensuring efficient

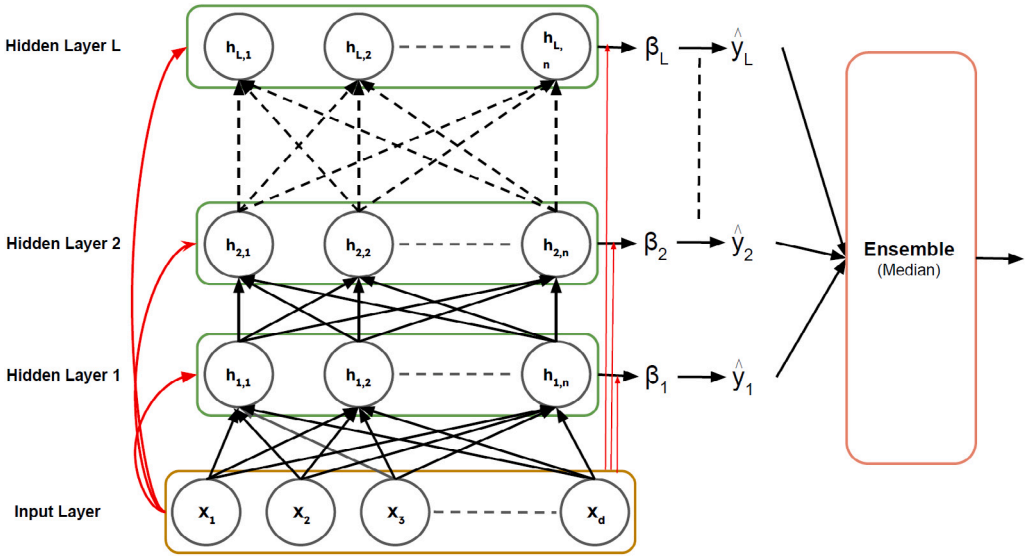

Fig. 3. Architecture of the edRVFL network.

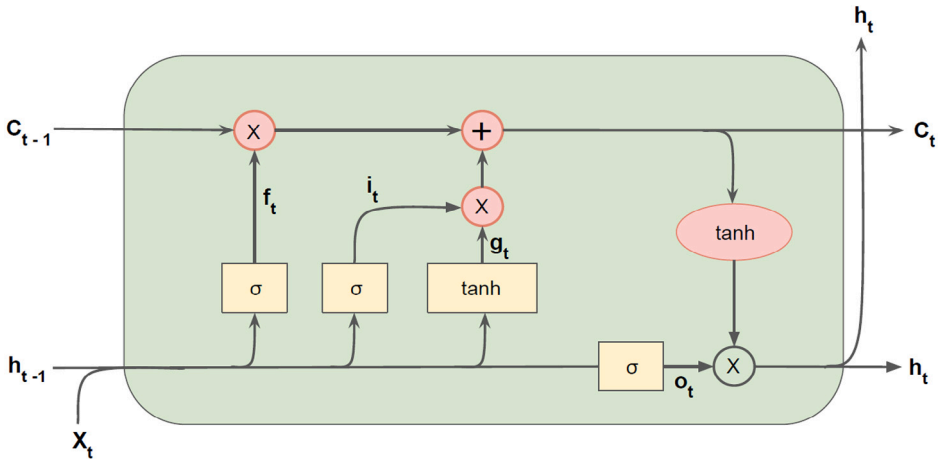

Fig. 4. Architecture of the LSTM network.

handling of prolonged dependencies within sequential data. The LSTM cell (Fig. 4) is composed of a gating mechanism, and each gate is tasked with overseeing and controlling the information flow within the memory cells. The input gate i_t manages the extent to which information is allowed into the memory cell, forget gate f_t manages information retention in the memory cell, and the output gate o_t determines the extent of information from the memory cell is passed to the next hidden state using the current input and prior hidden state.

$$i_t = \sigma(W_{Xi}X_t + W_{hi}h_{t-1}) \tag{8}$$

$$f_t = \sigma(W_{Xf}X_t + W_{hf}h_{t-1}) \tag{9}$$

$$o_t = \sigma(W_{Xo}X_t + W_{ho}h_{t-1}) \tag{10}$$

The cell state, C_t, is based on input, forget, and candidate cell state g_t. The candidate cell state and cell state are computed as follows:

$$g_t = \tanh(W_{Xg}X_t + W_{hg}h_{t-1}) \tag{11}$$

$$C_t = f_t \cdot C_{t-1} + i_t \cdot g_t \tag{12}$$

Then, the output, current hidden state h_t, is calculated by employing the output gate to process the cell state through the tanh activation function.

$$h_t = o_t \cdot \tanh(C_t) \tag{13}$$

2.4. Bayesian optimization for hyperparameter tuning

Bayesian optimization (BO) stands out as an advanced technique employed in hyperparameter tuning, utilizing a surrogate function to describe the conditional probability of efficacy on a validation set [26]. Unlike traditional grid search methods, BO efficiently retains and utilizes all previous computations, thus avoiding redundant evaluations of suboptimal hyperparameter configurations. The algorithm integrates an acquisition function to systematically identify the most promising hyperparameter configuration for evaluation in subsequent iterations.

The Bayesian optimization algorithm (BOA) comprises five core components: the surrogate function, the hyperparameter search space, the acquisition function, the objective function, and the historical record of evaluations. Within this framework, the objective function is characterized by the predictive accuracy of the validation set. The surrogate function is constructed through the tree-based Parzen window estimation (TPE) algorithm, and the chosen acquisition function is the expected improvement. The formulation of the acquisition function, denoted as $S_{f_o^\star}(v)$, is expressed as follows:

$$S_{f_o^\star}(v) = \int_{-\infty}^{f_o^\star} (f_o^\star - f_o) P(f_o|v) df_o \tag{14}$$

Here, $f_o^\star$ is the objective function, and f_o signifies the predetermined threshold for the objective function concerning the hyperparameters υ. The TPE algorithm encompasses the subsequent steps, explained in Algorithm 1.

Algorithm 1 TPE method for Hyperparameter Tuning

Input: Objective function f_o, TPE method $\mathcal{M}$, hyperparameter space $\mathbb{H}_\upsilon$, acquisition function validation S, initialized memory $\mathcal{D}$.
Output: Best hyperparameters $\upsilon^\star$ in memory $\mathcal{D}$.

1: **for** $i \leftarrow 1$ to N **do**
2: $P(f_o|\upsilon) \leftarrow$ Fit memory $\mathcal{D}$ using $\mathcal{M}$
3: Maximize acquisition function S in Equation (14) to find the next hyperparameter choice υ_{i+1}
4: Evaluate the objective function $f_o(\upsilon_{i+1})$
5: Update memory $\mathcal{D}$: $\mathcal{D} \leftarrow \mathcal{D} \cup (\upsilon_{i+1}, f_o(\upsilon_{i+1}))$
6: **end for**

3. Proposed methodology

This section primarily addresses the proposed network. Subsequently, we will focus on the training algorithm, followed by a discussion of the testing algorithm for the proposed model. Lastly, we will discuss the hyperparameter tuning setup for the proposed model.

3.1. Recurrent ensemble deep random vector functional link network (RedRVFL)

This paper introduces an innovative approach called the Recurrent ensemble deep random vector function link (RedRVFL) network, aimed at augmenting the capabilities of the existing edRVFL by integrating recurrent connections to capture temporal and sequential behaviours. While the edRVFL has proven effective for handling tabular datasets, it falls short in capturing the nuanced sequential patterns inherent in time series data. As a result, its application for forecasting future trends in time series datasets is impractical. To address these limitations, we present the RedRVFL network, designed to enhance the edRVFL's performance by incorporating recurrent connections to model the sequential nature of time series data adeptly. The idea behind our proposed model architecture originates from a gap identified in the existing literature concerning sequential behaviour detection utilizing randomized networks, particularly the edRVFL variant. The proposed model consists of multiple hidden layers stacked on each other with a skip connection to the input layer. Each layer processes input sequences with the help of LSTM. The output of one layer serves as the input to the next, allowing for hierarchical learning of complex temporal patterns. Moreover, the RedRVFL hidden layer, which comes after the first layer, also utilizes information from the raw input data with the help of a skip connection for better representation. This integration equips the RedRVFL network with the ability to capture long-term dependencies, thereby enabling accurate predictions of future trends in time series datasets. The RedRVFL algorithm decomposes the task into multiple smaller tasks by initializing the hidden weights to be random and kept fixed and computing output weights independently for each layer. The network comprises a total of L hidden layers, with N hidden nodes within each layer, akin to LSTM latent units. The structure of the RedRVFL network having L hidden layers is illustrated in Fig. 1.

Let X denote the input data with dimensions $R^{n\times t_l\times d}$, where n is the number of observations, t_l signifies the temporal order or time lag, and d indicates the number of feature dimensions in the time series forecasting model. Reshaping the input data to $R^{n\times p}$, where $p = t_l \times d$, initiates the processing stage. The input data X undergoes the first hidden layer comprised of LSTM unit in an unrolling way. The resulting output, denoted as h^1_{RC} and derived from Eq. (13), possesses dimensions of $R^{n\times N}$.

$$h^1_{RC} = f_l(W^{(1)}_R; X) \tag{15}$$

Here, f_l represents the non-linearity function obtained from the LSTM hidden layer. In other words, $f_l(W^{(1)}_R; X)$ signifies the randomly initialized weights $W^{(1)}_R$ associated with the hidden layer comprising LSTM cells and the input data X. The LSTM hidden layer processes the input X, yielding a hidden output denoted as h^1_{RC} obtained from Eq. (13).

Subsequently, the feature dataset X is concatenated with the output h^1_{RC}, forming a combined representation denoted as $[h^1_{RC}X]$. This combined representation maintains dimensions of $R^{n\times(p+N)}$. This processing and concatenation procedure repeats for the following hidden layers, where the outputs from the previous layers and the original features serve as inputs. The outputs from the subsequent hidden layers are computed as follows, similar to Eq. (16):

$$h^l_{RC} = f_l(W^{(i)}_R; [h^{(i-1)}_{RC}, X]) \tag{16}$$

Additionally, the output weights β_l corresponding to lth hidden layers can be computed using the Eqs. (4) and (5), employing Ridge regression as the chosen method for weight computation. The loss function for the lth hidden layer is formulated as:

$$\mathcal{L}_l = \min_{\beta_l} \|[h^l_{RC}\ X]\beta_l - Y\|^2 + \lambda\|\beta_l\|^2 \tag{17}$$

Here, β_l represents the output weight vector corresponding to the lth layer, with λ serving as the regularization parameter. The output weight values follow the Eqs. (4) and (5) for primal and dual space, respectively.

In the augmented feature matrix $D = [h^l_{RC}\ X]$, which concatenates the input and hidden features, the final output at the lth layer can be calculated as:

$$\hat{y}_l = [h^l_{RC}\ X]\beta_l \tag{18}$$

3.2. Training

The training algorithm for the RedRVFL can be summarized in Algorithm 2. Firstly, the hyperparameters, including the hidden layer dimension, the number of hidden layers, and the regularization strength, are specified. Following this, the hidden layer weights undergo random initialization, and a feed-forward computation is performed using Eqs. (6) or (7) to evaluate the hidden state.

Later, the training algorithm concludes by yielding the output layers' weights, denoted as $\beta = [\beta_1, \beta_2, \ldots, \beta_L]$. These output weights incorporate the learned information from each layer and can be utilized for various downstream tasks, including decision-making or regression, depending on the specific application context. The training algorithm provides a systematic and iterative process for optimizing the network's weights and achieving desirable performance on the given task.

3.3. Testing

The final forecast generated by the RedRVFL model is constructed as an ensemble of outputs of each hidden layer. To combine these forecasts effectively, various forecast combination approaches can be employed. The efficiency and robustness of mean and median in forecasting are highlighted in recent research by Wang et al. [27]. As discussed in the study, these aggregation methods provide reliable forecasts by effectively summarizing data and minimizing the impact of outliers. Wang et al.'s comprehensive review underscores the effectiveness of mean and median in capturing underlying trends and patterns in time series data. This paper adopts the median as the combination operator in our proposed RedRVFL model.

Algorithm 2 RedRVFL training algorithm

Input: N hidden latent LSTM units, number of layers L, regularization parameter λ.
Output: $\beta = [\beta_1, \beta_2, \dots, \beta_L]$
Randomly initialize the weights $W_R^{(1)}, W_R^{(2)}, \dots W_R^{(l)}$
for $l \leftarrow 1{:}L$ **do**
  if $l = 1$ **then**
    Calculate h_{RC}^1 using $W_R^{(1)}$ from Equations (13) and (15)
    Calculate β_1 using λ from Equation (4) or (5)
  else
    Calculate h_{RC}^l using $W_R^{(l)}$ from Equations (13) and (16)
    Calculate β_l using λ from Equation (4) or (5)
  end if
end for

The entire testing process for the RedRVFL model is outlined in Algorithm 3. Upon completing the training process, the RedRVFL model computes the predicted output, denoted as $\hat{y}$, through a feed-forward mechanism. By employing the ensemble approach and utilizing the median as the combination operator, the RedRVFL model leverages the collective wisdom of multiple outputs, resulting in an enhanced forecast that demonstrates improved performance. The values are preset and treated as constants throughout the testing method. Typically, validation data is used to hyper-parameter tune these parameters. This guarantees that the hidden outputs are normalized and distributed throughout training.

Algorithm 3 encompasses the steps involved in generating predictions using the RedRVFL model. It incorporates the essential operations, from initializing the network and training it with the available data to computing the final forecast through the ensemble approach, incorporating the individual outputs from all layers. The RedRVFL model ensures a robust and efficient testing process that produces accurate and reliable predictions by adhering to this algorithmic procedure.

Algorithm 3 RedRVFL testing algorithm

Input: The hidden layer weights W_R, output layer weights β, regularization parameter λ
Output: $\hat{y}$
for $l \leftarrow 1{:}L$ **do**
  if $l = 1$ **then**
    Compute h_{RC}^1 using $W_R^{(1)}$ from Equations (13) and (15)
    Compute output $\hat{y_1}$ using Equation (18)
  else
    Compute h_{RC}^l using $W_R^{(l)}$ from Equations (13) and (16)
    Compute $\hat{y_l}$ using Equation (18)
  end if
end for
  Compute the final output $\hat{y}$ by taking the median of $\hat{y_1}, \dots \hat{y_L}$

3.4. Hyper-parameter tuning

Hyperparameter tuning is crucial as it optimizes the performance and generalization ability of deep learning models. Layerwise parameter tuning offers advantages over traditional tuning methods by allowing for a more nuanced adjustment of model parameters, which can enhance convergence and overall model efficiency, especially in complex neural network architectures. The experimental setup used for the proposed model involves the same hyper-parameter setup as used in [20]. The hyper-parameters for the RedRVFL are as follows: the number of latent units for LSTM, N, the number of hidden layers, L, the regularization parameter λ, and the input scale.

In the RedRVFL network, the output from each hidden layer is propagated to the subsequent hidden layer, enabling the extraction of advanced features. Notably, each hidden layer represents a regressor and is associated with a distinct set of hyperparameters specific to that layer. Moreover, each layer's hyperparameters depend on the preceding layer's hyperparameters. Consequently, the tuning of hyperparameters occurs based on the layerwise Bayesian optimization technique and is kept fixed, and then the hyperparameter for the next layer is fined-tuned. This approach ensures a systematic and sequential refinement of the model's hyperparameters throughout the network.

4. Experiments

This section describes an empirical investigation conducted on eleven financial time series sourced from Yahoo Finance,[1] an Open Source platform. We provide a brief overview of the dataset and the pre-processing steps applied. Subsequently, evaluation metrics are defined. Finally, we describe the benchmark models and the experimental settings.

4.1. Dataset description

Table 1 summarizes the descriptive statistics for eleven financial time series datasets. These time series data comprise various index datasets and stock exchange datasets. The stock-indexed datasets are the following: Dow Jones Industrial Index (DJI), Hang Seng Index (HSI), Korean Composite Stock Price Index (KOSPI), RUSSELL 2000, National Association of Securities Dealers Automated Quotations Stock Exchange (NASDAQ), National Stock Exchange 50 (NIFTY50), Stock Exchange Sensitive Index (SENSEX), and Standard & Poor's 500 Index (S&P500). The stock exchange datasets are the London Stock Exchange (LSE), New York Stock Exchange (NYSE), and Shanghai Stock Exchange (SSE) Composite Index. The data is collected daily from 2013 to 2022 and is publicly accessible through the official website of Yahoo Finance.[2]

The data pre-precessing approach ensures better outputs from the machine learning model. In deep networks, maintaining proper initialization of weights and biases is essential for prediction [28]. Constraining the range of weights and biases to $[0, 1]$ promotes diversity among neurons, aiding in the effective capture of complex patterns and preventing uniform feature acquisition [20]. This range facilitates diversity among neurons, thereby enhancing the ability to capture intricate patterns effectively while averting uniform feature acquisition. This paper uses max–min normalization as a pre-processing technique for the raw data. Let us consider that the maximum and minimum values of the datasets are denoted as x_{max} and x_{min}, respectively. Subsequently, the data undergoes a transformation to be constrained within the specified range of $[0, 1]$, accomplished through the utilization of the Equation:

$$x_{normalized} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{19}$$

The dataset undergoes division into three subsets: training, validation, and test data. Specifically, the validation set comprises 10% of the dataset, serving the purpose of fine-tuning the model during training, while the test set constitutes 20%. The partitioning considers time series behaviour; hence, validation data immediately follows the training data, and the test data follows the validation data. Following the training and fine-tuning phases, the training and validation data sets are combined to form the full training data set. Subsequently, testing is conducted using this consolidated data set.

[1] https://finance.yahoo.com/.
[2] https://finance.yahoo.com/.

Table 1
Dataset summary.

Dataset	Mean	Median	Min	Max	Skewness	Kurtosis	Range	Std.
DJI	23 426.381	23 441.759	13 104.139	36 799.648	0.372	−1.106	23 695.508	6527.001
HSI	24 698.745	24 358.269	14 687.019	33 154.121	0.021	−0.339	18 467.101	3181.866
KOSPI	2253.495	2099.490	1457.640	3305.209	1.285	0.825	1847.569	364.156
NASDAQ	7623.561	6867.359	3019.51	16 057.44	0.779	−0.567	13 037.93	3497.702
NIFTY50	10 601.953	10 092.35	5285.00	18 812.50	0.724	−0.453	13 527.50	3556.175
RUSSELL2000	1474.502	1426.550	849.35	2442.74	0.714	−0.292	1593.39	370.931
SENSEX	35 272.887	32 720.16	17 905.910	63 284.191	0.746	−0.482	45 378.281	12 086.134
S&P500	2742.172	2584.84	1426.189	4796.56	0.674	−0.650	3370.37	872.8410
LSE	4558.64	3893.50	1002.207	9910.00	0.364	−1.350	8907.793	2532.684
NYSE	12 283.167	11 928.629	8443.509	17 353.759	0.634	−0.515	8910.25	2188.937
SSE	3017.059	3090.635	1950.012	5166.35	0.052	0.827	3216.338	527.809

4.2. Evaluation metrics

This paper incorporates three error measures for evaluation purposes. Firstly, Root Mean Square Error (RMSE) is utilized, and it computes the square root of the mean of the squared variances between predicted and observed values. Another measure used is the mean absolute error (MAE), which provides a straightforward evaluation of the average absolute difference between predicted and actual values. MAE is not sensitive to outliers and provides a scale-dependent measure of model performance. The mean absolute percentage error (MAPE) serves as a measure to assess forecast accuracy within diverse time series data. It computes the average percentage difference between predicted and actual values, facilitating unbiased comparisons irrespective of scale variations. The specific definitions of these three measures are provided in the following Equations:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - f_i)^2} \tag{20}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - f_i| \tag{21}$$

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - f_i}{y_i}\right| \times 100 \tag{22}$$

Here, y_i represents the actual value, f_i represents the forecasted value, and n represents the total number of observations.

4.3. Compared models

In order to evaluate the effectiveness of our proposed model, we conducted a thorough assessment by comparing it against a number of existing benchmark models. This included Statistical, machine learning, deep learning models, and decomposition technique-based models. As the proposed model is a hybrid of both machine learning and deep learning algorithms, so we needed to compare it against state-of-the-art benchmarks in both domains for a comprehensive evaluation.

The compared models used are named as follows:

1. Autoregressive Integrated Moving Average (ARIMA) [1]
2. Persistence [29]
3. Support Vector Regression (SVR) [4]
4. Temporal Convolutional Neural Network (TCN) [30]
5. Long Short-Term Memory (LSTM) Network [31]
6. Gated Recurrent Unit (GRU) [32]
7. Variational Mode Decomposition LSTM (VMD-LSTM) [33].
8. Random Vector Function Link Network (RVFL) [34]
9. Empirical Wavelet Transform-RVFL (EWTRVFL) [35]
10. Ensemble deep echo state network (edESN) [36].
11. Ensemble deep RVFL (edRVFL) [20].
12. Empirical Wavelet Transform-edrvfl (EWTedRVFL) [20].

Table 2
Hyper-parameter search space for the benchmark models.

Model	Parameter	Values
ARIMA	p/q	[1, 2, 3]
	d	[0, 1]
SVR	C	$[2^{-10}, 2^{0}]$
	ϵ	[0.001, 0.01, 0.1]
	Radius	[0.001, 0.01, 0.1]
LSTM	Hidden nodes	[4, 8, 16, 32, 64]
	Layers	[1, 2, 3]
	Optimizer	Adam
	Activation	tanh
GRU	Hidden nodes	[4, 8, 16, 32, 64]
	Layers	[1, 2, 3]
	Optimizer	Adam
	Activation	tanh
TCN	Filters	[4, 8, 16, 32, 64]
	Kernel size	[1, 2, 3]
	Optimizer	Adam
	Activation	[sigmoid, tanh, ReLU]
edESN	Reservoir size	[20, 200]
	Spectral radius	[0.95, 1]
	Input scalings	[0, 1]
	Leaky rate	[0.95, 1]
	Transient	[1, 49]
	Regularization parameter	[0, 1]
RVFL-based models	Hidden nodes	[20, 200]
	Layers	[1, 12]
	Regularization parameter	[0, 1]
	Input scaling	[0, 1]
	k	[2, 3, 4]
	Window	[48, 96, 124]

4.4. Experimental settings

For a fair and thorough comparison, we fine-tuned the settings of all models using a cross-validation method. The specific values we explored during this fine-tuning process are detailed in Table 2, providing insights into how we optimized the models. It is worth noting that certain settings remained consistent across all models to maintain uniformity. These fixed settings include a batch size of 32, a learning rate of 0.001, and a set number of training epochs at 100. For VMD-LSTM [33], we adopted the parameters from the original paper, making necessary adjustments for a fair comparison.

In the case of RedRVFL, we took a strategic approach to optimization. Each layer underwent individual optimization using Bayesian optimization, involving 100 iterations for each layer. For example, if RedRVFL is configured with 10 layers, the total Bayesian optimization iterations would be 1000.

To ensure fairness and comparability across all models, we standardized the number of Bayesian optimization iterations for RVFL, EWTRVFL, edESN, edRVFL, EWTedRVFL, and RedRVFL. This standardization ensures that each model undergoes an equal optimization

Table 3
RMSE results.

Dataset	ARIMA [1]	Persistence [29]	SVR [4]	TCN [30]	LSTM [31]	GRU [32]	RVFL [34]	EWTRVFL [35]	VMD-LSTM [33]	edESN [36]	edRVFL [20]	EWTedRVFL [20]	RedRVFL†
DJI	2123.402	422.841	490.419	501.810	432.603	436.595	375.830	379.753	438.390	353.158	**342.784**	342.821	345.485
HSI	4369.332	369.522	685.370	5691.688	**353.231**	356.730	378.378	378.820	388.228	378.317	378.351	373.716	374.805
KOSPI	591.239	32.057	53.544	552.733	35.348	35.377	31.256	52.137	29.724	31.266	30.494	30.688	**30.515**
NASDAQ	1965.874	204.260	355.617	542.154	**184.003**	204.869	208.540	248.033	182.092	212.829	207.732	207.893	207.423
NIFTY50	1498.384	178.021	241.950	207.554	179.097	195.829	171.968	247.344	**157.208**	336.214	172.273	173.147	170.843
RUSSELL2000	226.980	32.971	51.536	38.483	33.086	39.374	32.273	37.786	35.594	43.253	32.027	**31.960**	32.160
SENSEX	4987.525	608.845	772.853	739.142	606.935	654.839	579.022	676.718	**535.616**	1124.155	579.597	591.038	578.746
S&P500	323.681	54.063	77.296	74.428	51.008	51.745	51.191	58.180	50.453	60.965	52.270	50.173	**50.165**
LSE	2244.193	152.899	256.863	196.461	164.800	163.838	142.864	145.099	148.142	144.734	141.895	145.879	**141.145**
NYSE	1054.741	197.376	229.789	226.688	197.330	238.223	173.338	179.689	206.635	392.097	**169.497**	170.292	171.312
SSE	225.251	36.033	44.736	35.027	37.336	36.028	33.588	33.662	40.815	35.206	33.506	33.626	**33.401**

Table 4
MAE results.

Dataset	ARIMA [1]	Persistence [29]	SVR [4]	TCN [30]	LSTM [31]	GRU [32]	RVFL [34]	EWTRVFL [35]	VMD-LSTM [33]	edESN [36]	edRVFL [20]	EWTedRVFL [20]	RedRVFL†
DJI	1851.642	290.014	411.655	400.781	280.529	289.431	296.002	293.599	291.208	270.596	**257.917**	258.025	260.992
HSI	3691.853	275.741	575.280	5386.436	**267.376**	268.293	291.449	287.657	298.647	285.074	291.407	282.396	286.844
KOSPI	487.465	24.558	43.750	529.566	27.923	26.650	24.338	43.367	21.988	24.336	23.909	24.144	**23.869**
NASDAQ	1701.543	152.941	306.098	456.414	138.670	165.005	159.789	196.998	140.026	164.049	159.300	161.094	159.338
NIFTY50	1282.927	129.108	196.400	167.893	130.113	152.149	131.165	199.065	**117.692**	256.671	131.523	132.506	129.925
RUSSELL2000	203.447	25.315	43.569	30.404	26.716	31.312	25.935	30.630	27.558	34.679	25.726	**25.674**	25.872
SENSEX	4269.155	439.651	628.069	598.377	**430.150**	493.717	438.811	524.567	386.950	850.952	438.867	451.186	439.500
S&P500	273.792	38.682	64.436	60.374	38.324	38.083	39.200	45.590	**36.537**	47.167	40.443	37.721	37.681
LSE	2137.535	107.245	222.421	152.261	113.851	113.404	**97.439**	98.541	103.280	98.709	100.407	99.108	98.467
NYSE	899.926	139.648	188.925	179.455	134.157	178.370	135.537	141.923	139.237	328.470	**130.879**	131.886	133.503
SSE	198.578	26.018	36.392	26.637	27.721	25.909	25.146	25.432	30.027	26.856	25.194	25.503	**25.064**

Table 5
MAPE results.

Dataset	ARIMA [1]	Persistence [29]	SVR [4]	TCN [30]	LSTM [31]	GRU [32]	RVFL [34]	EWTRVFL [35]	VMD-LSTM [33]	edESN [36]	edRVFL [20]	EWTedRVFL [20]	RedRVFL†
DJI	0.0546	0.0099	0.0127	0.0125	0.0102	0.0104	0.0089	0.0089	0.0105	0.0082	**0.0078**	**0.0078**	0.0079
HSI	0.1529	0.0118	0.0205	0.2874	0.0103	0.0103	0.0129	**0.0125**	0.0116	0.0125	0.0129	0.0124	0.0127
KOSPI	0.1906	0.0099	0.0182	0.2203	0.0112	0.0106	0.0088	0.0149	0.0088	0.0088	**0.0086**	0.0087	**0.0086**
NASDAQ	0.1229	0.0132	0.0296	0.0403	**0.0123**	0.0144	0.0124	0.0151	**0.0123**	0.0127	0.0124	0.0125	**0.0123**
NIFTY50	0.0751	0.0094	0.0117	0.0098	0.0104	0.0121	0.0080	0.0120	0.0094	0.0154	0.0080	0.0080	**0.0079**
RUSSELL2000	0.0975	0.0143	0.0168	0.0168	0.0145	0.0176	0.0127	0.0148	0.0154	0.0168	0.0127	**0.0126**	0.0127
SENSEX	0.0745	0.0095	0.0115	0.0104	0.0103	0.0115	**0.0079**	0.0094	0.0092	0.0152	**0.0079**	0.0081	0.0080
S&P500	0.0642	0.0106	0.0175	0.0152	0.0102	0.0103	0.0095	0.0110	0.0106	0.0114	0.0097	**0.0091**	**0.0091**
LSE	0.2864	0.0138	0.0158	0.0196	0.0145	0.0144	**0.0127**	0.0129	0.0131	0.0129	0.0131	0.0130	0.0128
NYSE	0.0560	0.0101	0.0116	0.0119	0.0099	0.0131	0.0086	0.0090	0.0106	0.0204	**0.0084**	**0.0084**	0.0085
SSE	0.0580	0.0080	0.0115	0.0085	0.0084	0.0079	**0.0075**	0.0076	0.0091	0.0080	**0.0075**	0.0076	**0.0075**

process, contributing significantly to the reliability and robustness of our comparative analysis.

5. Empirical results and discussion

This section provides an overview of the empirical results of comparing the proposed and existing models. We present a detailed analysis using RMSE, MAE, and MAPE evaluation metrics. Next, we discuss the computational complexity of both the proposed and compared models. Subsequently, the results will be discussed using a statistical test, namely Wilcoxon's test. This test will enhance understanding of the observed differences and significance among the evaluated models. By incorporating evaluation metrics and statistical testing, a comprehensive assessment of the proposed model's performance relative to existing models will be presented, allowing for a robust and thorough comparison.

The results of this evaluation are presented in Tables 3, 4, and 5. These tables provide a detailed overview of the performance metrics, including RMSE, MAE, and MAPE, allowing for a thorough analysis and comparison of the model's performance on the financial time series datasets.

The comparative analysis reveals that certain models perform best on specific time series datasets, as indicated by the numbers in bold. The proposed model consistently delivers excellent performance across the majority of datasets. In particular, the edRVFL, VMD-LSTM model demonstrates exceptional performance. The EWTedRVFL model also achieves remarkable accuracy compared to the other models. It is worth noting that in select cases, the VMD-LSTM and EWTedRVFL model excels due to its adeptness in employing decomposition techniques, while the edESN model leverages its Echo State Network (ESN) architecture effectively. Upon examination of the results presented in Tables 3, 4, and 5, as well as the dataset statistics in the Table 1, it becomes evident that our proposed model struggles to achieve superior performance on highly volatile data such as the SENSEX. However, when considering the overall efficacy across a wide range of volatilities, the RedRVFL model performs better. These findings highlight the effectiveness and competitiveness of the proposed model while recognizing the strengths exhibited by the decomposition methods in specific scenarios.

Visualization study

In this investigation, we conducted a comprehensive analysis using visualization techniques on two prominent datasets sourced from Yahoo Finance[3] — specifically, the Dow Jones Index (DJI) and S&P500 datasets. The visual depictions portrayed in Fig. 5 aim to delve deeply into the temporal patterns manifested in these datasets, presenting both the raw data and the predictions generated by the RedRVFL model on the corresponding test datasets.

This meticulous visual scrutiny significantly enhances our understanding of the effectiveness of the RedRVFL model in forecasting financial time series. The alignment observed between the model's predictions and the actual data emphasizes its proficiency in accurately predicting closing prices, bolstering our confidence in its predictive capabilities. Consequently, this investigation emerges as a meaningful exploration into the predictive prowess of the RedRVFL model, particularly within the realm of financial time series forecasting.

Computational time

Efficient computation time calculation is essential for evaluating algorithmic efficiency, informing resource allocation decisions, and gauging solution scalability in practical scenarios. The computational costs of the algorithms, encompassing both proposed and comparison models, have been meticulously outlined in Table 6, except the persistence model, which does not require training. The computation time (in sec) is computed by keeping the hyperparameter fixed on the DJI

[3] https://finance.yahoo.com/.

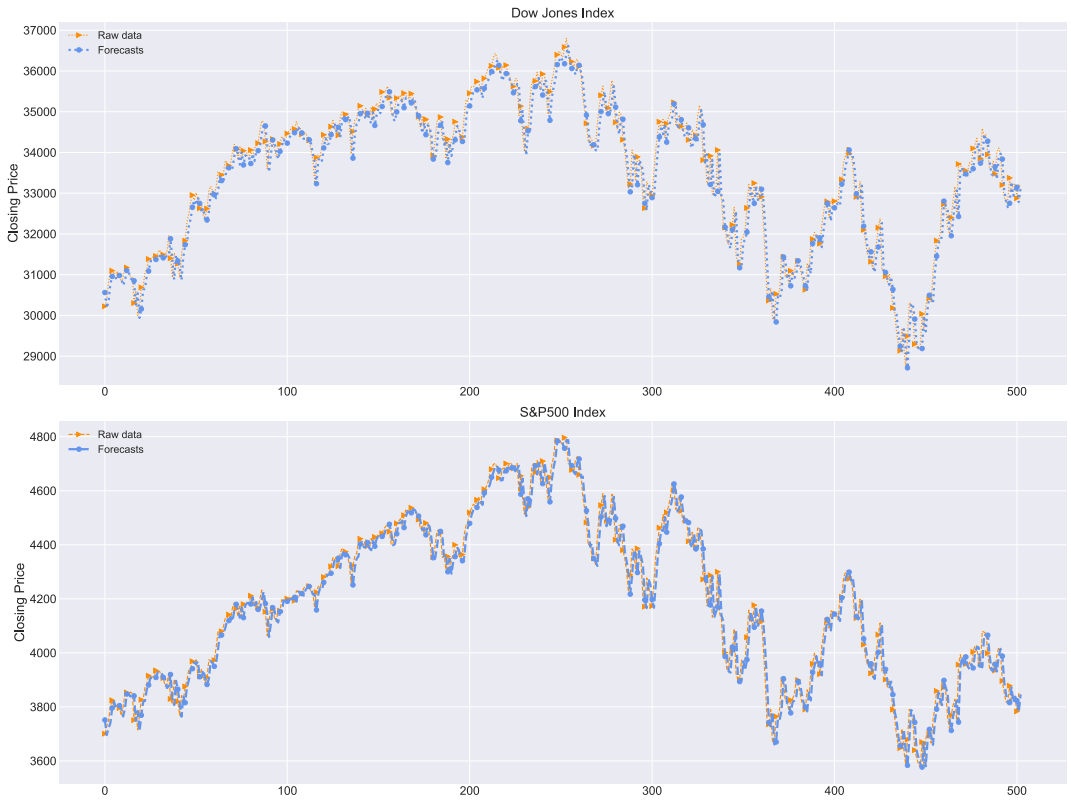

Fig. 5. Visualization of raw data and forecasts for DJI and S&P500 index dataset.

Table 6
The table illustrates the computational cost of the proposed model and compared baselines.

Model	ARIMA	SVR	TCN	LSTM	GRU	VMD-LSTM	RVFL	EWTRVFL	edESN	edRVFL	EWTedRVFL	Proposed
Time (in sec)	1.65	0.02	41.59	26.57	22.42	100.78	0.02	0.03	0.58	0.11	0.13	1.02

dataset and keeping all settings fixed for fair comparison. The VMD-LSTM algorithm takes the longest time to compute, while the RVFL algorithm takes the shortest. The results show that techniques based on randomization perform better in terms of computational efficiency compared to algorithms that need training, while techniques based on decomposition take more time than the base model. The results are evidence of the Randomization technique's superiority over the trained algorithms. The proposed model takes more time to compute than the base model. This is because it has more parameters, especially with the incorporation of LSTM.

Empirical study using Wilcoxon test

We conducted an empirical study employing the Wilcoxon test, an essential statistical tool for evaluating performance through comparative analysis. In this context, it is crucial to note that a lower rank signifies superior performance. The *p*-value, a key metric derived from the paired Wilcoxon test, provides valuable insights into the statistical significance of observed differences.

Tables 7, 8, and 9 present the outcomes of the statistical comparisons involving the proposed model: RedRVFL and other models used for comparison using RMSE, MAE, and MAPE respectively. These tables feature the average ranks, with the model achieving the lowest rank being considered the most favourable in terms of performance.

The proposed model consistently obtains a lower average rank across all tables, as shown in Tables 7, 8, and 9. This illustrates that the proposed model exhibits statistical superiority when contrasted with the comparative model.

Table 7
Statistical comparison between proposed and other models using Wilcoxon's test using RMSE.

Method	Avg. rank	p-value
RedRVFL†	**2.63**	
EWTedRVFL [20]	3.81	5e−1
edRVFL [20]	3.45	3e−1
edESN [36]	8.45	9e−4
VMD-LSTM [33]	5.45	6e−1
EWTRVFL [35]	7.90	9e−4
RVFL [34]	4.63	9e−4
GRU [32]	7.63	2e−2
LSTM [31]	6.18	8e−2
TCN [30]	10.36	9e−4
SVR [4]	11.18	9e−4
Persistence [29]	6.36	4e−1
ARIMA [1]	12.90	9e−4

6. Ablation study

The proposed model is introduced primarily utilizing the recurrent neural network, edRVFL network, and BOA hyper-parameter tuning. Then, the ablation study is conducted to validate the importance of each component. We have mainly defined three variants for the comparison, and these are as follows:

1. edRVFLBOA: This variant involves the fine-tuning of hyperparameters within the edRVFL network using the BOA algorithm.

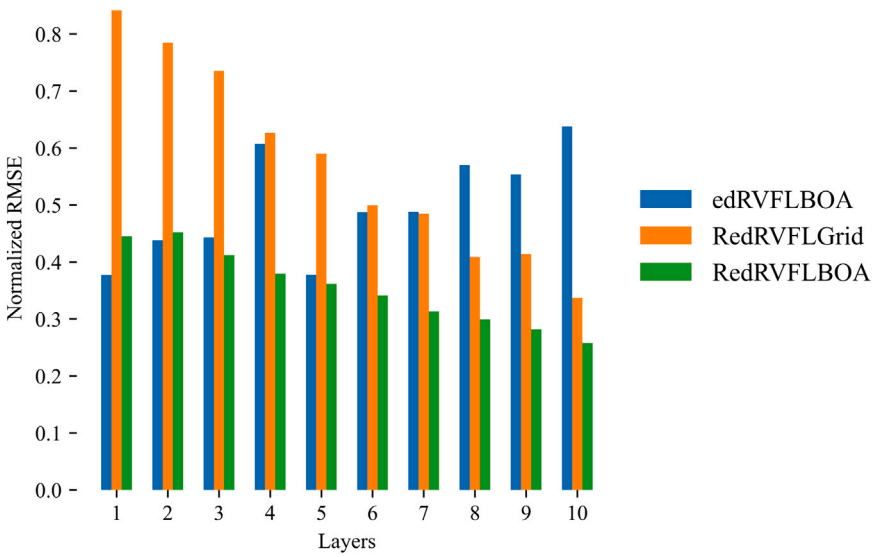

Fig. 6. The figure illustrates a bar plot illustrating normalized RMSE performance. The bars represent three different models: RedRVFLGrid, edRVFLBOA, and RedRVFLBOA.

Table 8
Statistical comparison between proposed and other models using Wilcoxon's test using MAE.

Method	Avg. rank	p-value
RedRVFL†	**3.27**	
EWTedRVFL [20]	4.36	6e−1
edRVFL [20]	4.27	7e−1
edESN [36]	8.36	4e−3
VMD-LSTM [33]	5.00	1e−2
EWTRVFL [35]	8.09	9e−4
RVFL [34]	5.36	5e−2
GRU [32]	7.18	1e−2
LSTM [31]	5.18	6e−1
TCN [30]	10.54	9e−4
SVR [4]	11.36	9e−4
Persistence [29]	5.18	1e−2
ARIMA [1]	12.81	9e−4

Table 9
Statistical comparison between proposed and other models using Wilcoxon's test using MAPE.

Method	Avg. rank	p-value
RedRVFL†	**2.63**	
EWTedRVFL [20]	3.54	4e−1
edRVFL [20]	3.27	3e−1
edESN [36]	8.00	4e−3
VMD-LSTM [33]	6.45	9e−3
EWTRVFL [35]	7.00	4e−3
RVFL [34]	4.00	2e−2
GRU [32]	8.45	1e−2
LSTM [31]	6.54	4e−2
TCN [30]	10.63	9e−4
SVR [4]	10.90	9e−4
Persistence [29]	6.72	4e−3
ARIMA [1]	12.81	9e−4

2. RedRVFLGrid: RedRVFL network employing grid search for hyperparameter tuning.
3. RedRVFLBOA: RedRVFL network utilizing the BOA algorithm for hyperparameter tuning.

The ablation study results are depicted in Fig. 6, illustrating the normalized error on the vertical axis to highlight distinctions among the variants. Initially, edRVFLBOA demonstrates strong performance, but as depicted in the figure, the proposed model consistently surpasses edRVFLBOA by a substantial margin in later layers. Notably, RedRVFLGrid exhibits poor performance in the initial layers but demonstrates improved performance compared to edRVFLBOA in subsequent layers. RedRVFLBOA, incorporating both BOA and weighted connections, emerges as the superior variant, emphasizing the necessity of BOA for enhanced performance, particularly beyond the initial layers. This is particularly evident in the layer-wise application of BOA, allowing for diverse configurations for each layer. The proposed model, integrating both recurrent connections and BOA, consistently outperforms all other variants, underscoring the recommendation to leverage both components for optimal edRVFL performance. This outcome highlights the inherent advantage of deep networks in extracting complex features from input data, emphasizing the potential of the proposed model across diverse applications.

7. Conclusion

This paper introduces an innovative Recurrent Ensemble Deep Random Vector Functional Link (RedRVFL) neural network for financial time series forecasting. The hidden layer weights are initialized randomly and remain constant throughout training. Output layer weights are determined using a closed-form solution. The hidden layers in this model are unsupervised and randomly initialized, and they utilize recurrent neural networks to identify sequential patterns within time series data. The model's hyperparameters are fine-tuned using layer-wise Bayesian optimization, with each hidden layer receiving input from the preceding layer. The final output is a combination of outputs from each layer. Notably, the proposed model offers a significant advantage over traditional deep learning techniques by leveraging randomization and avoiding the computational burden of backpropagation algorithms providing more practical applicability. Empirical results demonstrate the superiority of the proposed model, as evidenced by three error metrics and statistical tests.

The proposed model exhibits superiority for several reasons:

- The RedRVFL's architecture used the importance of both ensemble learning and recurrent networks. The RedRVFL treats each hidden layer as an individual predictor, and as a result, the ensemble of these reduces individuals' uncertainty.
- The proposed model's output layer adeptly captures linear patterns from the direct link and nonlinear patterns from the hidden layer features.

- The time series patterns are easily detected with the help of recurrent neural networks without losing the time series information in the data.

The empirical results demonstrate that our model performs better than the other models on the eleven financial time series datasets. The essential advantage of our model lies in its ability to efficiently detect time series patterns, exhibiting both efficacy and computational cost advantages. Future works could focus on enhancing the interpretability of the RedRVFL network, addressing its limitations when applied to highly volatile financial time series datasets. Additionally, exploring methodologies such as weighting averaging, meta-learning-based aggregation, and decomposition-based methods could offer valuable insights.

CRediT authorship contribution statement

Aryan Bhambu: Writing – review & editing, Writing – original draft, Software, Investigation. **Ruobin Gao:** Writing – review & editing, Supervision, Investigation. **Ponnuthurai Nagaratnam Suganthan:** Writing – review & editing, Supervision, Project administration, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgement

Open Access funding provided by the Qatar National Library.

References

[1] A.A. Ariyo, A.O. Adewumi, C.K. Ayo, Stock price prediction using the ARIMA model, in: 2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation, IEEE, 2014, pp. 106–112.
[2] P. Somani, S. Talele, S. Sawant, Stock market prediction using hidden Markov model, in: 2014 IEEE 7th Joint International Information Technology and Artificial Intelligence Conference, IEEE, 2014, pp. 89–92.
[3] F. Ye, L. Zhang, D. Zhang, H. Fujita, Z. Gong, A novel forecasting method based on multi-order fuzzy time series and technical analysis, Inform. Sci. 367 (2016) 41–57.
[4] K.-j. Kim, Financial time series forecasting using support vector machines, Neurocomputing 55 (1–2) (2003) 307–319.
[5] P. Meesad, R.I. Rasel, Predicting stock market price using support vector regression, in: 2013 International Conference on Informatics, Electronics and Vision, ICIEV, IEEE, 2013, pp. 1–6.
[6] B. Egeli, M. Ozturan, B. Badur, Stock market prediction using artificial neural networks, Decis. Support Syst. 22 (2003) 171–185.
[7] K.-j. Kim, W.B. Lee, Stock market prediction using artificial neural networks with optimal feature transformation, Neural Comput. Appl. 13 (2004) 255–260.
[8] T. Moudiki, F. Planchet, A. Cousin, Multiple time series forecasting using quasi-randomized functional link neural networks, Risks 6 (1) (2018) 22.
[9] P. Yu, X. Yan, Stock price prediction based on deep neural networks, Neural Comput. Appl. 32 (2020) 1609–1628.
[10] M. Alajanbi, D. Malerba, H. Liu, Distributed reduced convolution neural networks, Mesop. J. Big Data 2021 (2021) 25–28.
[11] A. Bhambu, Stock market prediction using deep learning techniques for short and long horizon, in: Soft Computing for Problem Solving: Proceedings of the SocProS 2022, Springer, 2023, pp. 121–135.
[12] Z. Han, J. Zhao, H. Leung, K.F. Ma, W. Wang, A review of deep learning models for time series prediction, IEEE Sens. J. 21 (6) (2019) 7833–7848.
[13] O.B. Sezer, M.U. Gudelek, A.M. Ozbayoglu, Financial time series forecasting with deep learning: A systematic literature review: 2005–2019, Appl. Soft Comput. 90 (2020) 106181.
[14] D. Ravì, C. Wong, F. Deligianni, M. Berthelot, J. Andreu-Perez, B. Lo, G.-Z. Yang, Deep learning for health informatics, IEEE J. Biomed. Health Inform. 21 (1) (2016) 4–21.
[15] S. Dörner, S. Cammerer, J. Hoydis, S. Ten Brink, Deep learning based communication over the air, IEEE J. Sel. Top. Sign. Proces. 12 (1) (2017) 132–143.
[16] J. Wang, T. Sun, B. Liu, Y. Cao, D. Wang, Financial markets prediction with deep learning, in: 2018 17th IEEE International Conference on Machine Learning and Applications, ICMLA, IEEE, 2018, pp. 97–104.
[17] G. Zhang, B.E. Patuwo, M.Y. Hu, Forecasting with artificial neural networks:: The state of the art, Int. J. Forecast. 14 (1) (1998) 35–62.
[18] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[19] L. Zhang, P.N. Suganthan, A survey of randomized algorithms for training neural networks, Inform. Sci. 364 (2016) 146–155.
[20] R. Gao, L. Du, P.N. Suganthan, Q. Zhou, K.F. Yuen, Random vector functional link neural network based ensemble deep learning for short-term load forecasting, Expert Syst. Appl. 206 (2022) 117784.
[21] Y.-H. Pao, Y. Takefuji, Functional-link net computing: theory, system architecture, and functionalities, Computer 25 (5) (1992) 76–79.
[22] J.C.A. Barata, M.S. Hussein, The Moore–Penrose pseudoinverse: A tutorial review of the theory, Braz. J. Phys. 42 (2012) 146–165.
[23] A.E. Hoerl, R.W. Kennard, Ridge regression: Biased estimation for nonorthogonal problems, Technometrics 12 (1) (1970) 55–67.
[24] P.N. Suganthan, On non-iterative learning algorithms with closed-form solution, Appl. Soft Comput. 70 (2018) 1078–1082.
[25] Q. Shi, R. Katuwal, P.N. Suganthan, M. Tanveer, Random vector functional link neural network based ensemble deep learning, Pattern Recognit. 117 (2021) 107978.
[26] M. Pelikan, D.E. Goldberg, E. Cantú-Paz, et al., BOA: The Bayesian optimization algorithm, in: Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99, Vol. 1, Citeseer, 1999.
[27] X. Wang, R.J. Hyndman, F. Li, Y. Kang, Forecast combinations: An over 50-year review, Int. J. Forecast. 39 (4) (2023) 1518–1547.
[28] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.
[29] S. Makridakis, S.C. Wheelwright, R.J. Hyndman, Forecasting Methods and Applications, John wiley & sons, 2008.
[30] S. Bai, J.Z. Kolter, V. Koltun, An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, 2018, arXiv preprint arXiv:1803.01271.
[31] K. Chen, Y. Zhou, F. Dai, A LSTM-based method for stock returns prediction: A case study of China stock market, in: 2015 IEEE International Conference on Big Data (Big Data), IEEE, 2015, pp. 2823–2824.
[32] K. Sako, B.N. Mpinda, P.C. Rodrigues, Neural networks for financial time series forecasting, Entropy 24 (5) (2022) 657.
[33] H. Niu, K. Xu, W. Wang, A hybrid stock price index forecasting model based on variational mode decomposition and LSTM network, Appl. Intell. 50 (2020) 4296–4309.
[34] Y. Ren, P.N. Suganthan, N. Srikanth, G. Amaratunga, Random vector functional link network for short-term electricity load demand forecasting, Inform. Sci. 367 (2016) 1078–1093.
[35] R. Gao, L. Du, K.F. Yuen, P.N. Suganthan, Walk-forward empirical wavelet random vector functional link for time series forecasting, Appl. Soft Comput. 108 (2021) 107450.
[36] R. Gao, R. Li, M. Hu, P.N. Suganthan, K.F. Yuen, Dynamic ensemble deep echo state network for significant wave height forecasting, Appl. Energy 329 (2023) 120261.

Aryan Bhambu received his B.Sc. degree from Rajasthan University, India, M.Sc. from Indian Institute of Technology Mandi, India. Currently, he is a Ph.D. student in the Department of Mathematics in Indian Institute of Technology Guwahati, India. His research interest includes time series forecasting, deep neural networks, statistical learning.

Ruobin Gao received his BEng degree from Jilin University, China,M.Sc. and Ph.D. degrees from Nanyang Technological University, Singapore. He is a research fellow at Nanyang Technological University. His research interests are time series forecasting and statistical learning. He also has extensive research experience in fuzzy time series and echo state neural network.

Ponnuthurai Nagaratnam Suganthan (or P.N. Suganthan) received the B.A degree, Postgraduate Certificate and M.A degree in Electrical and Information Engineering from the University of Cambridge, UK in 1990, 1992 and 1994, respectively. He received an honorary doctorate (i.e. Doctor Honoris Causa) in 2020 from University of Maribor, Slovenia. After completing his Ph.D. research in 1995, he served as a pre-doctoral Research Assistant in the Dept. of Electrical Engineering, University of Sydney in 1995–96 and a lecturer in the Dept of Computer Science and Electrical Engineering, University

of Queensland in 1996–99. He is currently a Research Professor at Qatar University. He is an Editorial Board Member of the Evolutionary Computation Journal, MIT Press (2013–2018). He is an associate editor of the IEEE Trans. on Cybernetics (2012 – 2018), IEEE Trans. on Evolutionary Computation (2005 –), Information Sciences (Elsevier) (2009 –), Pattern Recognition (Elsevier) (2001 –), Applied Soft Computing (2018–), Neurocomputing (2018–) and Engineering Applications of Artificial Intelligence (2022 –) Journals. He is a founding co-editor-in-chief of Swarm and Evolutionary Computation (2010 –), an SCI Indexed Elsevier Journal. His co-authored SaDE paper (published in April 2009) won the ‘‘IEEE Trans. on Evolutionary Computation outstanding paper award’’ in 2012. His research interests include swarm and evolutionary algorithms, pattern recognition, big data, deep learning and applications of swarm, evolutionary & machine learning algorithms. He was selected as one of the highly cited researchers by Thomson Reuters every year from 2015 to 2023 in computer science. He served as the General Chair of the IEEE SSCI 2013. He has been a member of the IEEE since 1991 and Fellow since 2015. He was an elected AdCom member of the IEEE Computational Intelligence Society (CIS) in 2014–2016. He is an IEEE CIS distinguished lecturer (2018–2021).