Information Sciences 547 (2021) 1066–1079

Contents lists available at ScienceDirect

Information Sciences

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

Advantages of direct input-to-output connections in neural networks: The Elman network for stock index forecasting

Yaoli Wang [a], Lipo Wang [b,*], Fangjun Yang [c], Wenxia Di [d], Qing Chang [e]

[a] College of Information and Computer, Taiyuan University of Technology, 030600, China
[b] School of Electrical and Electronic Engineering, Nanyang Technological University, 639798, Singapore
[c] College of Information and Computer, Taiyuan University of Technology, 030600, China
[d] Foreign Languages Department, Taiyuan Normal University, 030600, China
[e] College of Information and Computer, Taiyuan University of Technology, 030600, China

ARTICLE INFO

Article history:
Received 15 April 2020
Received in revised form 21 August 2020
Accepted 17 September 2020
Available online 28 September 2020

Keywords:
Direct input-to-output connections (DIOCs)
The Elman neural network
Stock index forecasting

ABSTRACT

The Elman neural network (ElmanNN) is well-known for its capability of processing dynamic information, which has led to successful applications in stock forecasting. In this paper, we introduce direct input-to-output connections (DIOCs) into the ElmanNN and show that the proposed Elman neural network with DIOCs (Elman-DIOCs) significantly out-performs the original ElmanNN without such DIOCs. Four different global stock indices, i.e., the Shanghai Stock Exchange (SSE) Composite Index, the Korea Stock Price Index (KOSPI), the Nikkei 225 Index (Nikkei225), and the Standard & Poor's 500 Index (SPX), are used to demonstrate the affecacy of the Elman-DIOCs in time-series prediction. We systematically evaluate 8 models, depending whether or not there are hidden layer biases, whether or not there are output layer biases, and whether or not there are DIOCs. The experimental results show that DIOCs lead to much better prediction accuracy, while requiring fewer than a half of the hidden neurons. Take the SPX index, for example - the root mean squared error (RMSE) and the mean absolute error (MAE) of the Elman-DIOCs are improved by 44.2% and 41.1%, respectively, compared to the ElmanNN, and 65.6% and 60.8%, respectively, compared to the multi-layer perceptron (MLP). We argue that (1) DIOCs can always help to improve accuracy, while reducing network complexity and computational burden, as long as the problem at hand (either regression or classification) has linear components, and (2) most real-world applications contain linear components. Therefore DIOCs will be almost always beneficial in any types of neural networks for classification or regression. We also point out that in rare cases where the problem at hand is entirely nonlinear, DIOCs should not be used.

1. Introduction

Artificial neural networks [38,39] are capable of learning from sample data to approximate any nonlinear functions with arbitrary precision [3,6,28,42,47]. In particular, the Elman recurrent neural network (ElmanNN) [11], with feedback loops in its hidden layer, has shown great promise in time-series analysis. For example, considering speech signals time-series, Achanta and Gangashetty [1] studied deep ElmanNNs for statistical parametric speech synthesis. They showed that deep

* Corresponding author.
E-mail addresses: wangyaoli@tyut.edu.cn (Y. Wang), elpwang@ntu.edu.sg (L. Wang), wendy_cn@263.net (W. Di).

https://doi.org/10.1016/j.ins.2020.09.031
0020-0255/

ElmanNNs were better suited for acoustic modeling compared to deep belief neural networks and performed competitively compared to deep long short-term memory (LSTM) neural networks, gated recurrent units (GRU), and simplified LSTMs. Ao and Palade [2] employed ElmanNNs for temporal modeling of microarray continuous time- series data. The prediction results on simulated non-stationary datasets and real biological datasets outperformed other existing approaches. Jibin Wang [36] added convolutional layers at the input of an ElmanNN and used this novel network to differentiate various kinds of life-threatening arrhythmias. Experimental results showed that this approach led to higher accuracy compared to a variety of other methods, such as the LSTM, the convolutional neural network (CNN), a combination of the CNN and the LSTM, discrete state Markov models, random forests, and multilayer perceptron (MLP) neural networks. Krishnan et al. [17] collected EEG signals and processed the signals using the hyper analytic wavelet transform with adaptive noise cancellation. They presented ElmanNN-based data protection, cryptography and authentication.

To carry out multi-step electricity price prediction, Rani and Victoire [30] proposed an enhanced hybrid framework, integrating the refined variational mode decomposition and the group search optimization algorithm for training the ElmanNN. The variational mode decomposition method was optimized using complement particle swarm optimization, so as to decompose the non-stationary pricing data into an optimum number of intrinsic mode functions. They showed that the ElmanNN out-performed other approaches. Kolanowski et al. [16] presented a navigation system based on an ElmanNN and data fusion from different sensors. The data samples to train the ElmanNN were collected during the test flight of a Quadrocopter. Hongtao Li et al. [19] used an ElmanNN optimized by a cuckoo search algorithm to forecast air cargos. The air cargo time-series from three different airports in China were adopted to validate the performance of the proposed approach and the empirical results showed that the proposed method was superior to all other benchmark models in terms of robustness and accuracy. Jalal et al. [15] used the ElmanNN and the NARX Neural Network for forecasting call volumes in call centers. The results could help determine the optimal number of agents necessary to reduce waiting time for customers, maximize profit, and minimize costs. The experimental results indicated that the proposed method was more efficient in forecasting call volumes compared to other methods.

Jiangyu Zheng [49] applied an ElmanNN in forecasting opening prices of the Shanghai Stock Exchange. Binghui Wu and Tingting Duan used the ElmanNN to predict stock [43] and gold future markets [44]. As discussed above, there are many different types of time-series for which the ElmanNN can enjoy useful applications. In this paper, we shall propose an improved ElmanNN and select stock market indexes as test examples.

Prediction of stock time-series has always been a hot and yet difficult research topic. Stocks play an important role in the financial industry. Stock market fluctuations will affect not only the economic system, but also the vital interests of investors. Therefore, it has become a major exploration direction to predict stock prices with a reasonable accuracy. With the rapid development in artificial intelligence, several techniques, such as neural networks, expert systems[10,4], and support vector machines[25,45,24,22,50], have gradually become mainstream models for stock prediction.

Tai-liang Chen and Feng-yu Chen [9] proposed a traditional pattern recognition model, based on PIP bull-flag pattern matching and floating-weighted bull-flag templates, to recognize bull-flag stock patterns. A bull-flag pattern is a turning point of a stock price. Their experimental results indicated that the proposed model outperformed the rough set theory (RST), genetic algorithms (GAs) and a hybrid model, as well as earlier work on traditional pattern recognition models [35]. Mu-Yen Chen and Bo-Tsuen Chen [8] proposed a novel fuzzy time-series model to forecast stock market prices. The proposed model was verified using experimental datasets from various stock indexes, and results were compared against existing fuzzy time-series models, three different SVM models, and three modern economic models. Compared to other current forecasting methods, their proposed models provided improved prediction accuracy and their results were verified by paired two-tailed t-tests, although no comparisons were made with neural networks.

Xiaoxiang Guo et al. [13] proposed a Delay Parameterized Method (DPM) for low dimensional mid-term chaotic time-series forecasting. They used particle swarm optimization and genetic algorithms [20,34,41] to obtain optimal parameters for prediction. They applied their approach to forecasting stock K-line maps, among other time-series, and obtained high quality predictions. This method would need to assume that stock prices have chaotic properties, which is yet to be rigorously proven.

Shekhar Gupta and Lipo Wang [14] utilized MLP neural networks to forecast future index prices of the NASDAQ 100 and the Standard & Poor 500. The effects of training the network with the most recent data, together with gradually sub-sampled past index data, were studied. They were able to obtain significantly higher returns compared to earlier work. Moews et al. [23] used an MLP neural network that employed step-wise linear regressions with exponential smoothing in the preparatory feature engineering for stock directional change prediction. During their experiments, the number of hidden layers in the neural network and the number of neurons in each layer were carefully adjusted to achieve the best accuracy. Laboissiere et al. [18] forecasted daily maximum and minimum stock prices of three Brazilian power distribution companies using an MLP neural network, together with attribute selection by correlation analysis. Ramezanian et al. [29] applied an integrated framework consisting of genetic network programming along with an MLP neural network to forecast the stock return. The results derived from 9 stocks in the Tehran Stock Exchange Market were better compared to the ARMA-GARCH model.

Yu Fang et al. [12] improved stock market prediction using wavelet neural networks (WNN) [33] and reported significantly better accuracies compared to existing approaches to stock market prediction. The prediction system was tested using Shenzhen Composite Index data. Mahmud and Meesad [21] proposed a novel approach for time-series stock market price prediction using a recurrent neuro-fuzzy system with momentum. Four top-listed stocks from the Dhaka stock exchange were applied to demonstrate the model strength.

Ren et al. [31] systematically evaluated effects of direct input-to-output connections (DIOCs) and neuron biases on the random vector functional link (RVFL) [26], a single hidden layer neural network in which the hidden neuron parameters are randomly chosen and the output weights are determined by pseudo-inverse or Moore–Penrose generalized inverse, for power load forecasting. The experimental results showed that the DIOCs had significant impact on forecasting accuracy and the results were statistically significant. DIOCs were also been added to the MLP neural networks trained with back-propagation (BP) and were shown to improve forecasting accuracy for short-term load [27] and other time-series [40]. Inspired by the above work, we introduce direct input-to-output connections (DIOCs) into the Elman neural network (ElmanNN) in this paper. Four stock indices, i.e., the SSE, the KOSPI, the Nikkei225, and the SPX, are used to compare the prediction performance of both the proposed ElmanNN-DIOCs and the original ElmanNN without DIOCs.

2. The proposed method

The Elman neural network is a recurrent network introduced by Elman in 1990 [11] and is composed of four layers, i.e., the input layer, the hidden layer, the undertaking layer, and the output layer, as shown in Fig. 1. The undertaking layer can be considered as a delay operator, which is used to store the output of the hidden layer. The input of the hidden layer is determined by both the input layer and the output of the undertaking layer.

$$x_t = f(w^{(1)}u_t + w^{(2)}x_t^{(c)} + b^{(h)}), \tag{1}$$

$$x_t^{(c)} = x_{t-1}, \tag{2}$$

where x_t is the output vector of the hidden layer, $w^{(1)}$ is the weight matrix from the input layer to the hidden layer, $w^{(2)}$ is the weight matrix from the hidden layer to the undertaking layer, u_t is the network input vector, $x_t^{(c)}$ is the output vector of the undertaking layer, $b^{(h)}$ is the hidden neuron bias vector, t represents the sampling time, and

$$f(x) = \frac{1}{1+e^{-x}}. \tag{3}$$

A linear transfer function is selected for the output layer neurons. The corresponding output is as follows

$$y_t = w^{(3)}x_t + b^{(o)}, \tag{4}$$

where $w^{(3)}$ is the weight matrix from the hidden layer to the output layer, $b^{(o)}$ is the output neuron bias vector, and y_t denotes the output vector of the network.

We propose an Elman model that has direct connections from the input layer to the output layer, as shown in Fig. 2. Thus the output vector, corresponding to Fig. 2, is as follows,

$$y_t = w^{(3)}x_t + \beta u_t + b^{(o)}, \tag{5}$$

where β represents the direct weight matrix from the input layer to the output layer.

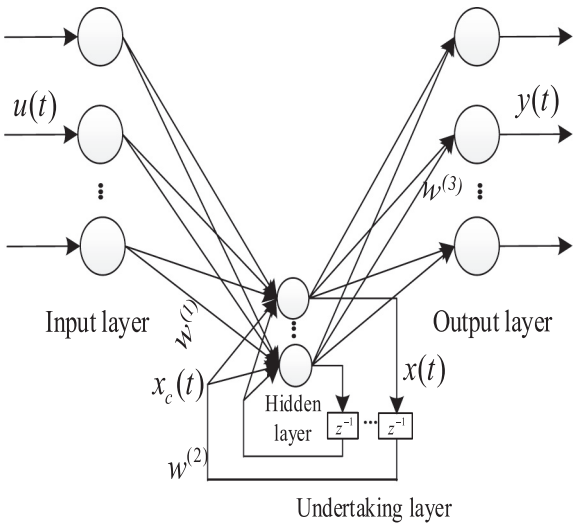


Fig. 1. The Elman neural network structure.

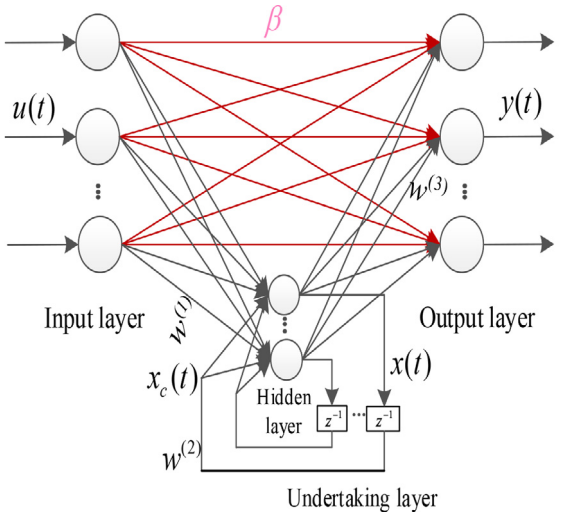


Fig. 2. Our proposed Elman neural network structure with direct input-to-output connections (DIOCs).

Compared with the standard Elman network shown in Fig.1, the input-to-output mapping of the proposed network shown in Fig. 2 adds a linear component. The elements of the weight matrix from the input layer to the output layer are first randomly assigned in the range of [0,1], and the proposed network utilizes the BP algorithm to optimize weights in the network. The network error function is defined as,

$$E = \frac{1}{2}\sum_{t=1}^{N}(o_t - y_t)^2, \tag{6}$$

where o_t is the target (objective) output at time t, y_t is the actual (predicted) output at time t, and N is the number of data points. In the following sections, effects of the direct input-to-output connections will be evaluated.

3. Experiment settings

3.1. Eight models

In order to compare the performance of the proposed network and obtain the best configuration, similar to how Ren et al. [31] discussed the RVFL, we systematically consider the presence and the absence of the hidden layer biases (the biases of the hidden neurons, termed "input layer biases" in [31]), the presence and the absence of the output layer biases (the biases of the output neurons, termed "hidden layer biases" in [31]), and the presence and the absence of the direct input-to-output connections (DIOCs), resulting in 8 models as shown in Table 1. M1, M3, M5, and M7 are networks with DIOCs.

3.2. Data selection and preprocessing

The daily closing index values of the Shanghai Stock Exchange (SSE) Composite Index, the Korea Stock Price Index (KOSPI), the Nikkei 225 Index (Nikkei225), and the Standard & Poor's 500 Index (SPX) [46] are selected as experimental datasets. The total number of data points in every dataset is 2000. The SSE data is selected from 10/12/2005 up to 31/12/2013, KOSPI from

Table 1
The 8 Elman neural network models with different configurations.

Model	Hidden layer biases	Output layer biases	DIOCs
M1	√	√	√
M2	√	√	×
M3	√	×	√
M4	√	×	×
M5	×	√	√
M6	×	√	×
M7	×	×	√
M8	×	×	×

13/12/2005 up to 30/12/2013, Nikkei225 from 09/11/2005 up to 30/12/2013, and SPX from 20/01/2006 to 31/12/2013, as is shown in Table 2.

We normalize data into $[0, 1]$ using the following equation:

$$S' = \frac{S - S_{min}}{S_{max} - S_{min}}, \tag{7}$$

where S is the raw data, and S_{min} and S_{max} represent the minimum and maximum values in each data set, respectively.

3.3. Training and prediction

Each dataset is divided into a training set and a testing set, i.e., the first 75% for training and the rest 25% for testing. Future closing prices are predicted using past closing prices. The normalized financial time-series is $\{y_1, y_2, \ldots, y_n, y_{n+1}, \ldots, y_t\}$, in which t represents the current moment. The predicted value y_{t+h} is

$$y_{t+h} = p(y_t, y_{t-1}, \ldots, y_{t-n+1}), \tag{8}$$

where h is the forecasting step size, p stands for the forecasting model, and n is the time delay that represents a window of past values used for prediction.

Constructed dataset samples are exhibited in Table 3. The input vector of the forecasting model contains the closing prices in n consecutive days, and the output is the predicted closing price after h trading days.

3.4. Performance measures

To assess the performance of a given model, we use the normalized root mean squared error (*RMSE*), the mean absolute error (*MAE*), and the mean absolute percentage error (*MAPE*) given as follows:

$$MAE = \frac{1}{N}\sum_{t=1}^{N}|o_t - y_t|, \tag{9}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(o_t - y_t)^2}, \tag{10}$$

and

$$MAPE = 100 \times \frac{1}{N}\sum_{t=1}^{N}\left|\frac{o_t - y_t}{o_t}\right|. \tag{11}$$

MAE, *RMSE*, and *MAPE* are used to measure the deviation between the predicted values and the target (true) values. The prediction performance is better when the values of these evaluation criteria are smaller. *MAPE* is often the recommended criterion for industrial practitioners.

4. Results and analyses

4.1. Preliminary results for the eight models

In order to compare the predictive performance of the eight models, extensive experiments are carried out with the SSE, KOSPI, Nikkei225, and SPX indices. The maximum training iterations is 3000 and the training error threshold is 0.0001. The optimal values for h and n as defined in Table 3 are 1 and 5, respectively. Table 4 shows examples of average *RMSE* of prediction results in 20 experiments. As shown in Table 4, the networks with DIOCs, i.e., M1, M3, M5, and M7 defined in Table 1, obtain lower *RMSE* (higher predictive accuracy) for the four financial indices, compared with other models without DIOCs, i.e., M2, M4, M6, and M8 defined in Table 1. Hence the DIOCs improve the models' predictive performance. Fig. 3 depicts the predicted results by M3 vs. the actual data for the SSE, KOSPI, Nikkei225, and SPX indices. The curves of the actual data and the predicted data are close.

4.2. The Wilcoxon signed-rank test

From Table 3, it is clearly seen that the DIOCs have favorable effects on network performance. In this sub-section, we use the Wilcoxon signed-rank test to verify the statistical significance of the performance differences in the 8 models, as Ren et al. [31] analyzed the RVFL.

In order to investigate whether the hidden layer biases have a significant effect on the network performance, the Wilcoxon signed-rank test is applied by comparing M1, M2, M3, and M4, with M5, M6, M7, and M8. The p values are recorded

Table 2
Datasets used in our studies.

Index	Duration	Data points
SSE	10/12/2005—31/12/2013	2000
KOSPI	13/12/2005—30/12/2013	2000
Nikkei225	09/11/2005—30/12/2013	2000
SPX	20/01/2006—31/12/2013	2000

Table 3
Constructed input–output data samples.

Output	Input
y_{n+h}	$y_1, y_2, \ldots, y_{n-1}, y_n$
y_{n+h+1}	$y_2, y_3, \ldots, y_n, y_{n+1}$
$\ldots$	$\ldots$
y_{t-1}	$y_{t-h-n}, y_{t-h-n+1}, \ldots, y_{t-h-2}, y_{t-h-1}$
y_t	$y_{t-h-n+1}, y_{t-h-n+2}, \ldots, y_{t-h-1}, y_{t-h}$

Table 4
Average testing *RMSE* for the 8 models defined in Table 1.

	M1	M2	M3	M4	M5	M6	M7	M8
SSE	30.90	39.93	29.54	39.75	27.89	30.40	27.50	29.01
KOSPI	18.59	21.82	17.52	20.39	17.65	18.34	17.56	17.94
Nikkei225	187.99	232.58	183.61	234.47	184.72	199.80	181.20	187.11
SPX	23.58	36.83	19.98	39.78	14.48	31.20	13.10	29.59

in Table 5 with the alternative hypothesis being 'two-sided' and $p < 0.05$, which is highlighted in bold and indicates a statistically significant difference. Table 5 shows that, with some cases, the hidden layer biases make a difference on the network performance, and therefore, the hidden layer biases should be retained in the network.

The Wilcoxon signed-rank test is also utilized to evaluate the impact of output layer biases by comparing M1, M2, M5, and M6, with M3, M4, M7, and M8. The p values are tabulated in Table 6 with the alternative hypothesis being 'two sided' and $p < 0.05$, which is highlighted in bold and indicates a statistically significant difference. The results reveal that the predicting performance differences are insignificant except for M6 vs. M8 on Nikkei225. Therefore, the output layer biases may be omitted in the network.

The outcomes of M1 vs. M2, M3 vs. M4, M5 vs. M6, and M7 vs. M8 reflect the effects of the DIOCs. From p values presented in Table 7, it can be seen that the DIOCs affect the performance of network significantly, due to the fact that all the p values are less than 0.05, as highlighted in bold.

Based on the above-mentioned observations, M3 has the overall best performance and the DIOCs improve the forecasting performance of the network. In addition, the hidden layer biases are beneficial and should be retained, while it does not matter whether the output layer biases are kept or not. These optimal network configurations, in terms of DIOCs, the hidden layer biases, and the output biases, are consistent with previous studies on the RVFL for time-series prediction [31], the RVFL for classification [48], and the MLP-DIOCs for time-series prediction [27,40] (A neural network equivalent to the RVFL without DIOCs was first proposed by Schmidt et al. [32] in 1992, where the hidden neuron parameters are randomly chosen and the output weights are determined by pseudo-inverse or Moore–Penrose generalized inverse).

4.3. Comparisons with the MLP

The proposed M3 is compared with the standard ElmanNN (also known as M4 in Table 1) and the MLP [37] on the SSE, KOSPI, Nikkei225, and SPX datasets. The total length of data selected are the same for different methods, the data size of training set and testing set are also the same, as described in Sections 3.2 and 3.3. The optimal parameters are obtained by repeated experiments. Table 8 presents the optimal numbers for different methods. Table 8 shows that much more hidden layer neurons are required in the original ElmanNN without DIOCs and the MLP [37] to achieve the best performance, compared with the proposed ElmanNN-DIOCs (M3).

Fig. 4 depicts the results of predicted values of testing sets for the four stock indices and shows that our proposed ElmanNN-DIOCs (M3) is the best, compared with the MLP [37] and the original ElmanNN without DIOCs.

The forecasting performance of different methods are assessed by RMSE, MAE, and MAPE. In Table 9, each experiment was run 20 times independently and the average is shown. We can see that for all the financial time-series, the predicted results of the proposed ElmanNN-DIOCs (M3) are significantly better than the original ElmanNN without DIOCs and the MLP [37]. For example, for the SPX index, the RMSE and MAE of M3 are decreased by 44.2% and 41.1%, respectively, compared with the

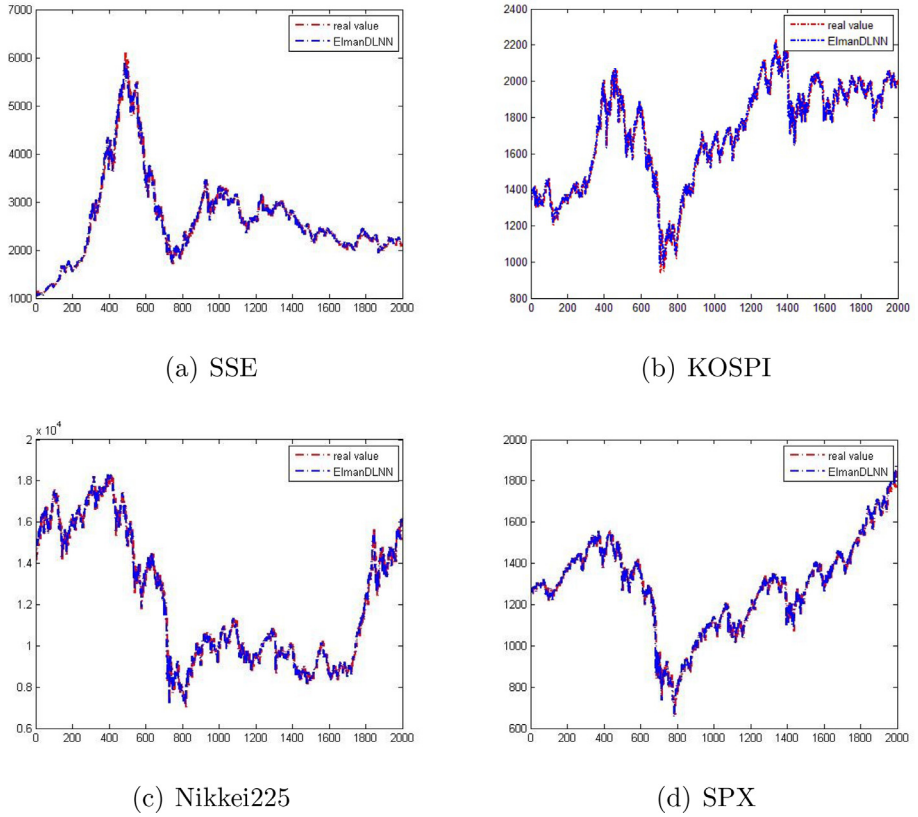


Fig. 3. The training and testing results by model M3 defined in Table 1 vs. the actual data for (a) the Shanghai Stock Exchange (SSE) Composite Index from 10/12/2005 up to 31/12/2013, (b) the Korea Stock Price Index (KOSPI) from 13/12/2005 up to 30/12/2013, (c) the Nikkei 225 Index (Nikkei225) from 09/11/2005 up to 30/12/2013, and (d) the Standard & Poor's 500 Index (SPX) from 20/01/2006 to 31/12/2013 [46]. The total number of data points in each index is 2000, 75% for training and the rest 25% for testing. In each sub-figure, the red curve represents the actual stock index data, while the blue curve is the output of the Elman neural network model, which shows that the Elman neural network is able to accurately model the index. Comparisons with other models are presented in Fig. 4, Table 8 and Table 9.

Table 5
Wilcoxon signed-rank test results for different Elman models with vs. without hidden layer biases. Boldfaces indicate statistically significant differences, whereas others represent no statistically significant differences.

	SSE	KOSPI	Nikkei225	SPX
M1 vs. M5	**3.36E−07**	0.0859	0.1806	**0.0021**
M2 vs. M6	**3.50E−06**	**4.17E−05**	**4.60E−04**	0.457
M3 vs. M7	0.0679	0.1199	0.1136	**0.004**
M4 vs. M8	**3.07E−06**	**3.75E−04**	**1.23E−07**	0.1075

Table 6
Wilcoxon signed-rank test results for different Elman models with vs. without output layer bias. Boldfaces indicate statistically significant differences, whereas others represent no statistically significant differences.

	SSE	KOSPI	Nikkei225	SPX
M1 vs. M3	0.0565	0.7764	0.072	0.2085
M2 vs. M4	0.6359	0.2616	0.8392	0.5979
M5 vs. M7	0.5792	0.8817	0.1075	0.5428
M6 vs. M8	0.1017	0.2085	**8.36E−04**	0.0601

original ElmanNN without DIOCs, and by 65.6% and 60.8%, respectively, compared with the MLP. The improvements are substantial for the other 3 indices.

Table 7
Wilcoxon signed-rank test results for different Elman models with vs. without direct input-to-output connections. Boldfaces indicate statistically significant differences, whereas others represent no statistically significant differences.

	SSE	KOSPI	Nikkei225	SPX
M1 vs. M2	**1.10E−05**	**4.60E−04**	**5.17E−06**	**0.0077**
M3 vs. M4	**3.99E−06**	**4.16E−04**	**1.92E−07**	**7.41E−05**
M5 vs. M6	**0.004**	**0.0071**	**5.63E−04**	**1.66E−07**
M7 vs. M8	**0.0223**	**0.0239**	**0.0066**	**6.80E−08**

Table 8
The optimal numbers of hidden neurons for different neural networks and stock indices.

	MLP	ElmanNN	Elman-DIOCs (M3)
SSE	16	14	6
KOSPI	14	16	6
Nikkei225	12	12	6
SPX	14	12	6

5. Discussions: Why Can Direct Input–Output Connections Help?

As mentioned in the Introduction, DIOCs were shown to help the MLP forecast short-term load [27] and other time-series [40], as well as benefit the RVFL in power load forecasting [31]. In addition to time-series forecasting, a regression problem, Zhang and Suganthan [48] also showed that DIOCs had a positive effect on the RVFL in classification through a comprehensive study of 121 UCI datasets [5,7]. These neural networks are all universal approximators without DIOCs and should therefore be able to adequately solve regression and classification problems without DIOCs. Why can DIOCs improve accuracy, sometimes substantially, while reducing network complexity and computational burden? Is this true for any data and any neural network architecture? Let us attempt to discuss these questions.

DIOCs basically provide a linear mapping from the input to the output if the output neurons have linear activation functions, for example, in most neural networks used for regression, as well as neural networks with random hidden neuron parameters and closed form solutions for the output layer weights, like the RVFL [26] and the Schmidt network [32].

In cases where the desired input-to-output mapping is entirely linear, although a neural network without DIOCs would be able to approximate such a linear mapping with arbitrary accuracy if given a sufficient number of hidden neurons, such a nonlinear neural network may be less accurate and less computationally efficient compared to a linear neural network. Even when theoretically there are sufficient hidden neurons for the required accuracy, the training outcome may not be ideal and the network parameters may be stuck in a local optimum, resulting in sub-optimal performance. In contrast, a neural network with DIOCs does not require any hidden neurons to model a linear input-to-output mapping.

On the other hand, if the desired input-to-output mapping is totally nonlinear, i.e., without any linear components, DIOCs would not help (their weights should vanish if training reaches a global optimum).

The results published so far on neural networks with DIOCs are based on somewhat real-world data. The fact that most of these results showed that the networks benefitted from DIOCs may mean that most of these real-world data require neither entirely linear input-to-output mapping, nor entirely nonlinear input-to-output mapping, but possess both 'linear and nonlinear components'. In these cases, we expect DIOCs would help in taking care of the linear components in the input-to-output mappings.

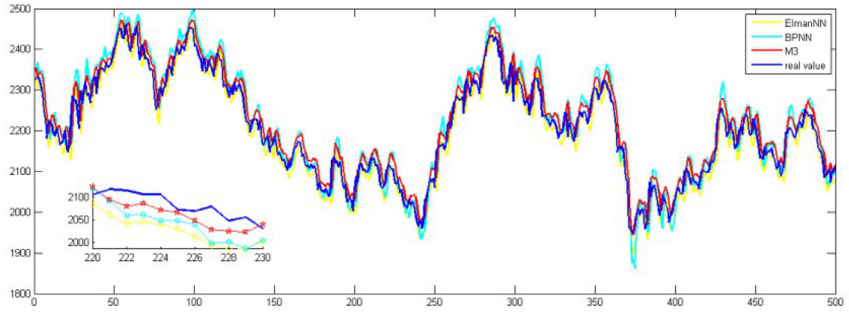
For the time-series prediction problem, a linear input-to-output mapping may be quite intuitive: it may be interpretted as naive trend-following. For example, in stock price prediction, an intuitive guess for the stock price for the next time step would be simply the same as the present stock price. If the output neurons have nonlinear activation functions, for example, in the multilayer perceptron (MLP), this trend-following interpretation may still hold if the activation functions of the output neurons are monotonic, like the sigmoid function given in Eq. (3).

Let us illustrate the above arguments using a toy example. The training input data are generated by:

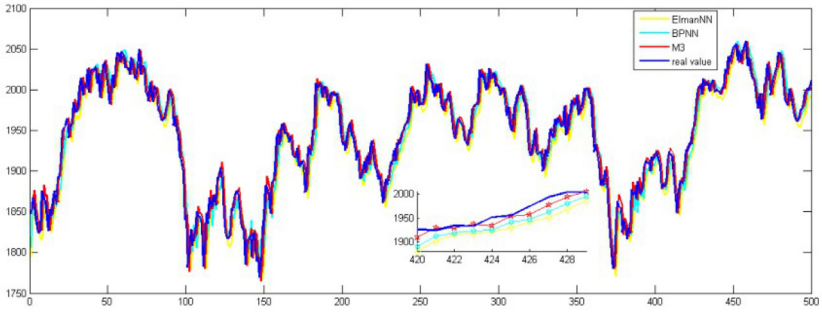
$$u_t = \sin\left(\frac{2\pi t}{20}\right), \tag{12}$$

where time step $t = 1, 2, \ldots$. Let us use an ElmanNN with 3 input nodes and 1 linear output neuron (Fig.1. We choose the target output to be:

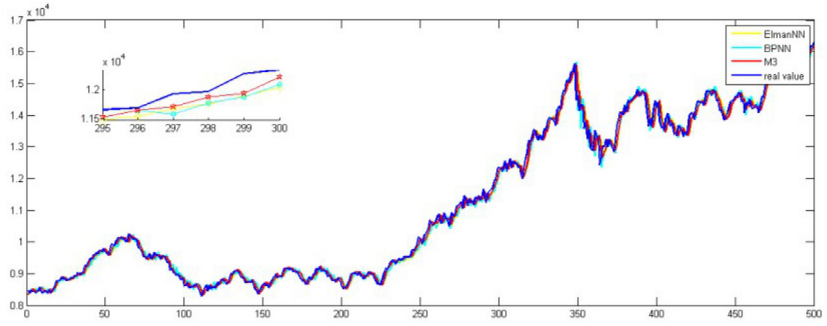
$$o_{t+1} = u_t + A\sin[10\ u_t]. \tag{13}$$



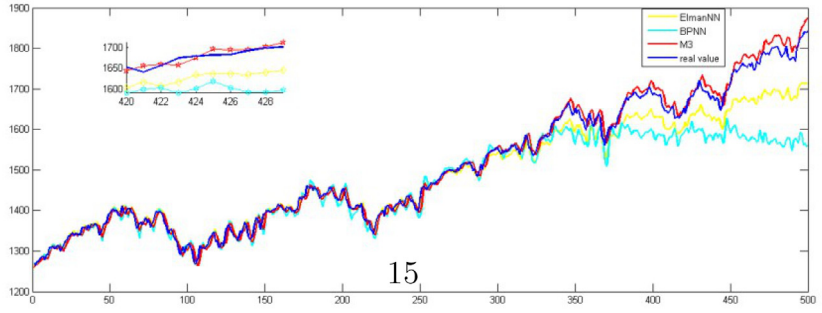
(a) SSE



(b) KOSPI



(c) Nikkei225



(d) SPX

Fig. 4. Comparisons of test results for the original Elman neural network (ElmanNN), the proposed Elman network with direct input-to-output connections (M3), and the multilayer perceptron network (BPNN or MLP), for the SSE, KOSPI, Nikkei225 and SPX.

Table 9
The prediction errors for the MLP, the original ElmanNN, and our Elman-DIOCs (M3).

Index	Errors	MLP	ElmanNN	Elman-DIOCs (M3)
SSE	RMSE	38.41	35.45	29.54
	MAE	30.72	28.11	22.83
	MAPE	0.59	0.51	0.40
KOSPI	RMSE	20.09	19.87	17.52
	MAE	15.63	15.30	13.42
	MAPE	0.11	0.19	0.09
Nikkei225	RMSE	209.37	208.46	183.61
	MAE	151.57	151.68	130.08
	MAPE	0.33	0.27	0.19
SPX	RMSE	58.15	35.83	19.98
	MAE	36.50	24.25	14.29
	MAPE	2.23	1.34	0.58

The data for the first 60 time steps were used for training and the data for the next 20 time steps were used for testing. The Levenberg–Marquardt training algorithm was selected. We used an error threshold 10^{-9}, but the training stopped before this error threshod could be reached because the training process converged. We compare the performance of the ElmanNN with and without DIOCs for different A values.

We first consider $A = 0$ (Fig. 5). In this case, we have

$$o_{t+1} = u_t, \tag{14}$$

i.e., the model prediction for output at time $t + 1$ is the naive choice of input at time t: an entirely linear input-to-output mapping. The network weights and testing RMSE for 1, 2, and 3 hidden neurons are shown in Table 10 (the weights from the input layer to the hidden layer $w^{(1)}$ and the feedback weights $w^{(2)}$ are unremarkable and omitted). Table 10 shows that in the presence of DIOCs, the weights from the hidden layer to the output neuron are roughly zero, indicating that the hidden neurons are not needed. In addition, the DIOCs are such that only the connection from the first input node to the output is roughly 1, whereas the other DIOCs are roughly zero.

In the absence of DIOCs, the test RMSE decreases as the number of hidden neurons increases, but is not as good as that in the presence of DIOCs for any number of hidden neurons (Fig. 6). We now consider $A = 0.1$ and $A = 10$ in Eq. (13), i.e., the input-to-output mapping has both linear and nonlinear components. Tables 11 and 12, as well as Figs. 7 and 8, show that DIOCs help to improve accuracy while reducing the number of hidden neurons needed when there are both linear and nonlinear components.

For a classification problem, a linear input-to-output mapping is less intuitive. The part of the network involving DIOCs is a (single-layer) Perceptron, which is able to solve linearly separable classification problems. Thus the fact that a classification network, such as the RVFL, also benefitted from DIOCs may mean that most of these real-world data are at least partially linearly separable. For an entirely nonlinearly separable problem, DIOCs should not be helpful or may even be harmful. For example, the XOR problem can be solved by an MLP with 2 input nodes, 2 hidden neurons, and 1 output neuron, without DIOCs. It seems unlikely to exist simpler networks that are able to solve the XOR problem and adding DIOCs would not be beneficial in this rather theoretical case.

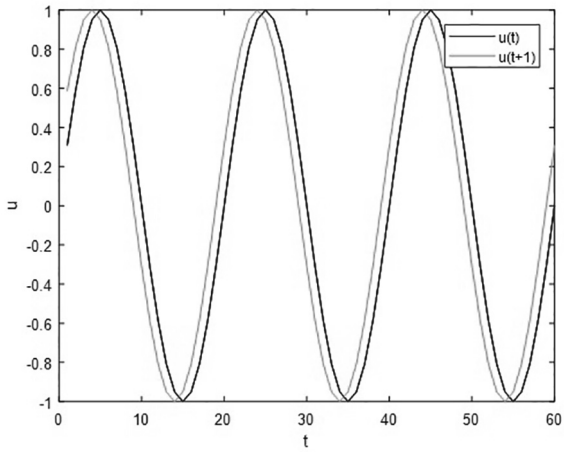


Fig. 5. Training input in the toy problem.

Table 10
Comparisons between the ElmanNN with vs. without DIOCs in a totally linear case: A = 0 in Eq. (13).

Number of hidden neurons	DIOCs	β	$w^{(3)}$	RMSE
1	√	−0.014; 0.027; 0.986	6.83e−4	1.96e−5
1	×		0.502	0.035
2	√	0.00; −0.00; 1.00	−0.00; 0.00	5.9e−6
2	×		−14.6; 30.3	3.15e−5
3	√	0.00; −0.01; 0.996	0.0001; 0.00; 0.00	2.77e−5
3	×		2.07; 5.22; −10.5	3.16e−5

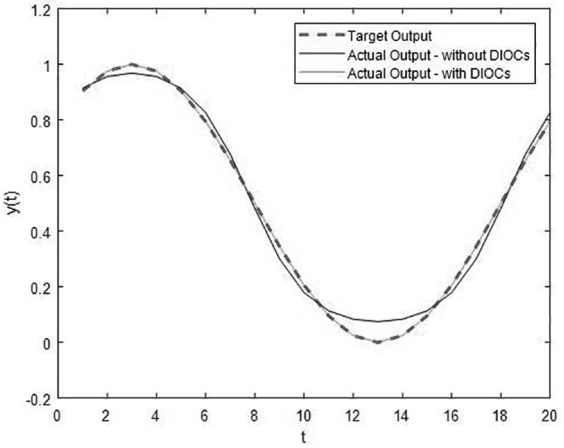


Fig. 6. Testing results for the ElmanNN with and without DIOCs in a totally linear case with 1 hidden neuron: A = 0 in Eq. (13).

Table 11
Comparisons between the ElmanNN with vs. without DIOCs when the input-to-output mapping has both linear and nonlinear components: A = 0.1 in Eq. (13).

Number of hidden neurons	DIOCs	β	$w^{(3)}$	RMSE
1	√	-0.0376; 0.0704; 1.0311	−0.0374	0.0263
1	×		1.095	0.0487
2	√	0.597; −2.024; 2.469	−0.149; 0.107	0.0123
2	×		−3.47; 2.28	0.0234
3	√	−1.35; 2.53; −0.099	−0.077; −0.069 0.065	0.017
3	×		0.14; 1.06; −1.45	0.018

Table 12
Comparisons between the ElmanNN with vs. without DIOCs when the input-to-output mapping has both linear and nonlinear components: A = 10 in Eq. (13).

Number of hidden neurons	DIOCs	RMSE
5	√	1.13e−5
5	×	0.02
10	√	1.6e−8
10	×	2.69e−6

6. Conclusions

Financial time-series prediction plays an important role in stock markets. In this study, we proposed a novel improved ElmanNN by introducing the direct input-to-output connections and showed experimentally that the direct input-to-output connections markedly improve prediction performance. Through extensive simulations using data from the SSE, KOSPI, Nikkei225, and SPX indices, we demonstrated that the enhancements are statistically significant. The predictive accuracy for all financial indices shows that our proposed ElmanNN-DIOCs outperforms the original ElmanNN without DIOCs and the MLP, while requiring fewer than half of the hidden neurons. We argue that (1) DIOCs can always help to improve

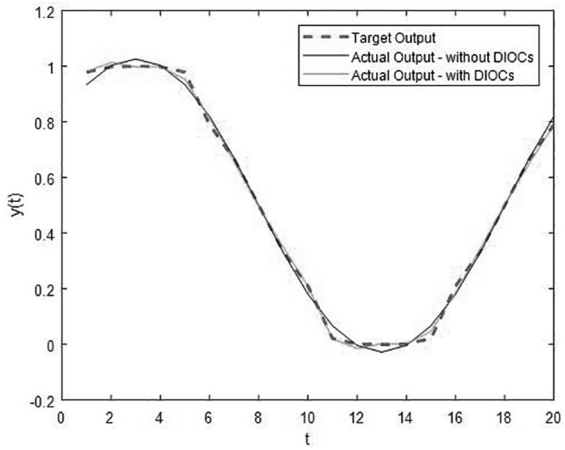


Fig. 7. Testing results for the ElmanNN with and without DIOCs in a mildly nonlinear case ($A = 0.1$ in Eq. (13)) with 2 hidden neurons.

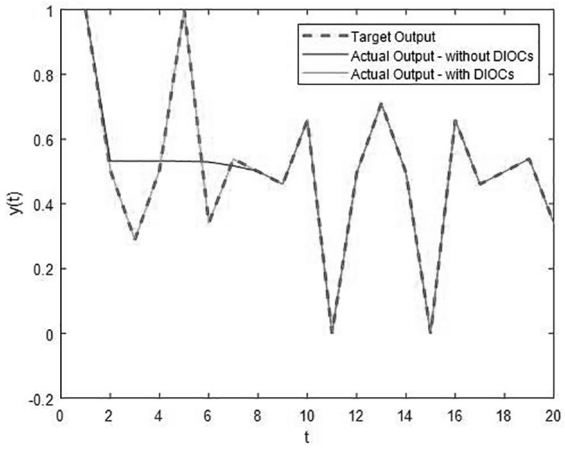


Fig. 8. Testing result for the ElmanNN with and without DIOCs in a strongly nonlinear case ($A = 10$ in Eq. (13)) with 5 hidden neurons.

accuracy, while reducing network complexity and computational burden, as long as the problem at hand (either regression or classification) has linear components, and (2) most real-world applications contain linear components, therefore DIOCs will be almost always beneficial. So far, advantages of direct input-to-output connections have been demonstrated in the Elman neural network, the RVFL, and the MLP. From our discussions in this paper, this phenomenon should likely exist for other types of neural networks for classification or regression, which will be subject to future studies.

CRediT authorship contribution statement

Yaoli Wang: Conceptualization, Methodology, Software, Supervision. **Lipo Wang:** Conceptualization, Software, Visualization, Funding acquisition, Validation. **Fangjun Yang:** Visualization, Investigation, Writing - original draft. **Wenxia Di:** Validation, Data curation, Writing - original draft. **Qing Chang:** Methodology, Data curation.

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.

Acknowledgements

This study is funded by Joint Research Fund for Overseas Chinese Scholars and Scholars in Hong Kong and Macao (Grant No. 61828601), Natural Science Foundation of Shanxi Province (Grant No. 201801D121141), and Provincial Program on Key Research Projects of Shanxi (Social Development Area) (Grant No. 201903D321003). We also thank the Editor-in-Chief and the reviewers for their constructive comments that helped to significantly improve this paper.

References

[1] S. Achanta, S.V. Gangashetty, Deep Elman recurrent neural networks for statistical parametric speech synthesis, Speech Commun. 93 (2017) 31–42.
[2] S.I. Ao, V. Palade, Ensemble of Elman neural networks and support vector machines for reverse engineering of gene regulatory networks, Appl. Soft Comput. 11 (2) (2011) 1718–1726.
[3] F. Beritelli, G. Capizzi, G.L. Sciuto, C. Napoli, M. Wozniak, A novel training method to preserve generalization of RBPNN classifiers applied to ECG signals diagnosis, Neural Networks 108 (2018) 331–338.
[4] M. Bildirici, E. Alp, Ö. Ersin, TAR-cointegration neural network model: An empirical analysis of exchange rates and stock returns, Expert Syst. Appl. 37 (2010) 2–11.
[5] C. Chen, K. Li, A. Ouyang, Z. Tang, K. Li, GFlink: an in-memory computing architecture on heterogeneous CPU-GPU clusters for big data, IEEE Trans. Parallel Distrib. Syst. 29 (6) (2018) 1275–1288.
[6] C. Chen, K. Li, S.G. Teo, X. Zou, K. Li, Z. Zeng, Citywide traffic flow prediction based on multiple gated spatio-temporal convolutional neural networks, ACM Trans. Knowl. Discovery Data (TKDD) 14 (4) (2020) 1–23.
[7] J. Chen, K. Li, K. Bilal, K. Li, P. Yu, A bi-layered parallel training architecture for large-scale convolutional neural networks in distributed computing environments, IEEE Trans. Parallel Distrib. Syst. 30 (5) (2019) 965–976.
[8] Mu-Yen Chen, Bo-Tsuen Chen, A hybrid fuzzy time series model based on granular computing for stock price forecasting, Inf. Sci. 294 (2015) 227–241.
[9] Tai-liang Chen, Feng-yu Chen, An intelligent pattern recognition model for supporting investment decisions in stock market, Inform. Sci. 346–47 (2016) 261–274.
[10] R. Dash, S. Samal, R. Dash, R. Rautray, An integrated TOPSIS crow search based classifier ensemble: In application to stock index price movement prediction, Appl. Soft Computing 85 (2019), Article 105784.
[11] J. Elman, Finding Structure in Time, Cognitive Sci. 14 (1990) 179–211.
[12] K. Yu Fang, L. Fataliyev, X.Fu. Wang, Y. Wang, Improving the genetic-algorithm-optimized wavelet neural network approach to stock market prediction, in: Proceedings of the 2014 International Joint Conference on Neural Networks, 2014, pp. 3038–3042.
[13] Xiaoxiang Guo, Yutong Sun, Jingli Ren, Low dimensional mid-term chaotic time series prediction by delay parameterized method, Inf. Sci. 516 (2020) 1–19.
[14] Shekhar Gupta, Lipo Wang, Stock forecasting with feedforward neural networks and gradual data sub-sampling, Australian J. Intell. Inform. Process. Syst. 11 (2010) 14–17.
[15] M.E. Jalal, M. Hosseini, S. Karlsson, Forecasting incoming call volumes in call centers with recurrent Neural Networks, J. Business Res. 69 (11) (2016) 4811–4814.
[16] K. Kolanowski, A. Aswietlicka, R. Kapela, J. Pochmara, A. Rybarczyk, Multisensor data fusion using Elman neural networks, Appl. Math. Comput. 319 (2018) 236–244.
[17] S. Krishnan, S. Lokesh, M.R. Devi, An efficient Elman neural network classifier with cloud supported internet of things structure for health monitoring system, Comput. Netw. 151 (2019) 201–210.
[18] L. Laboissiere, R. Fernandes, G. Lage, Maximum and minimum stock price forecasting of Brazilian power distribution companies based on artificial neural networks, Appl. Soft Comput. 35 (2015) 66–74.
[19] Hongtao Li, Juncheng Bai, Xiang Cui, Yongwu Li, Shaolong Sun, A new secondary decomposition-ensemble approach with cuckoo search optimization for air cargo forecasting, Appl. Soft Comput. 90 (2020) 106161.
[20] Y. Liu, N. Zhu, K. Li, M. Li, J. Zheng, K. Li, An angle dominance criterion for evolutionary many-objective optimization, Inf. Sci. 509 (2020) 376–399.
[21] M. Mahmud, P. Meesad, An innovative recurrent error-based neuro-fuzzy system with momentum for stock price prediction, Soft. Comput. 20 (2016) 4173–4191.
[22] I. Marković, M. Stojanović, J. Stanković, M. Stanković, Stock market trend prediction using ahp and weighted kernel ls-svm, Soft. Comput. 21 (18) (2017) 5387–5398.
[23] B. Moews, J. Herrmann, G. Ibikunle, Lagged correlation-based deep learning for directional trend change prediction in financial time series, Expert Syst. Appl. 120 (2019) 197–206.
[24] B. Nair, V. Mohandas, N. Sakthivel, A genetic algorithm optimized decision tree-svm based stock market trend prediction system, Int. J. Computer Sci. Eng. 02 (09) (2010) 2981–2988.
[25] R. Nayak, D. Mishra, A. Rath, A NaïveStojanović svm-knn based stock market trend reversal analysis for indian benchmark indices, Appl. Soft Comput. 35 (2015) 670–680.
[26] Y. Pao, G. Park, D. Sobajic, Learning and generalization characteristics of random vector functional-link net, Neurocomputing 6 (1994) 163–180.
[27] T. Peng, N. Hubele, G. Karady, Advancement in the application of neural networks for short-term load forecasting, IEEE Trans. Power Syst. 7 (1) (1992) 250–257.
[28] D. Polap, M. Wozniak, W. Wei, R. Damasevicius, Multi-threaded learning control mechanism for neural networks, Future Generation Comp. Syst. 87 (2018) 16–34.
[29] R. Ramezanian, A. Peymanfar, S. Ebrahimi, An integrated framework of genetic network programming and multi-layer perceptron neural network for prediction of daily stock return: An application in tehran stock exchange market, Appl. Soft Computing 82 (2019), Article 105551.
[30] R.H.J. Rani, T.A.A. Victoire, A hybrid Elman recurrent neural network, group search optimization, and refined VMD-based framework for multi-step ahead electricity price forecasting, Soft. Comput. 23 (2019) 8413–8434.
[31] Y. Ren, P. Suganthan, N. Srikanth, G. Amaratunga, Random vector functional link network for short-term electricity load demand forecasting, Inf. Sci. 367–368 (2016) 1078–1093.
[32] W.F. Schmidt, M. Kraaijveld, R.P. Duin, Feedforward neural networks with random weights, in: Proceedings of 11th IEEE International Conference on Pattern Recognition, 1992, pp. 1–4.
[33] K. Teo, L. Wang, Z. Lin, Wavelet packet multi-layer perceptron for chaotic time series prediction: effects of weight initialization, in: Proceedings of the 2001 International Conference on Computational Science, 2001, pp. 310–317.
[34] Z. Tong, H. Chen, X. Deng, K. Li, K. Li, A scheduling scheme in the cloud computing environment using deep Q-learning, Inf. Sci. 512 (2020) 1170–1191.
[35] Jar-Long Wang, Shu-Hui Chan, Stock market trading rule discovery using pattern recognition and technical analysis, Expert Syst. Appl. 33 (2) (2007) 304–315.
[36] Jibin Wang, Automated detection of atrial fibrillation and atrial flutter in ECG signals based on convolutional and improved Elman neural network, Knowl.-Based Syst. 193 (2020) 105446.
[37] Jian-Zhou Wang, Ju-Jie Wang, Zhe-George Zhang, Shu-Po Guo, Forecasting stock indices with back propagation neural network, Expert Syst. Appl. 38 (11) (2011) 14346–14355.

[38] L.P. Wang, X.J. Fu, Data Mining with Computational Intelligence, Springer-Verlag, Berlin/Heidelberg, 2005.
[39] L. Wang, K. Teo, Z. Lin, Predicting time series with wavelet packet neural networks, in: Proceedings of the 2001 IEEE International Joint Conference on Neural Networks, 2001, pp. 1593–1597.
[40] Y. Wang, L. Wang, Q. Chang, C. Yang, Effects of direct input-output connections on multilayer perceptron neural networks for time series prediction, Soft. Comput. 24 (2020) 4729–4738.
[41] M. Wozniak, K. Ksiazek, J. Marciniec, D. Polap, Heat production optimization using bio-inspired algorithms, Eng. Appl. AI 76 (2018) 185–201.
[42] M. Wozniak, D. Polap, Hybrid neuro-heuristic methodology for simulation and control of dynamic systems over time interval, Neural Networks 93 (2017) 45–56.
[43] B. Wu, T. Duan, A performance comparison of neural networks in forecasting stock price trend, Int. J. Comput. Intell. Syst. 10 (2017) 336–346.
[44] Wu. Binghui, Tingting Duan, The fractal feature and price trend in the gold future market at the Shanghai Futures Exchange (SFE), Physica A 474 (2017) 99–106.
[45] H. Xue, Y. Bai, Research on prediction of stock index based on pca and svm, in: Proceedings of the 2016 International Conference on Applied Mechanics, Mechatronics and Intelligent Systems, 2016, pp. 925–930.
[46] Yahoo. https://finance.yahoo.com/personal-finance.
[47] Jianhua Zhang, Jianrong Li, Rubin Wang Instantaneous mental workload assessment using time-frequency analysis and semi-supervised learning Cognitive Neurodynamics (online) 2020.
[48] L. Zhang, P. Suganthan, A comprehensive evaluation of random vector functional link networks, Inf. Sci. 367–68 (2016) 1094–1105.
[49] Jiangyu Zheng, Forecast of opening stock price based on Elman neural network, Chem. Eng. Trans. 46 (2015) 565–570.
[50] M. Zhu, L. Wang, Intelligent trading using support vector regression and multilayer perceptrons optimized with genetic algorithms, in: Proceedings of the 2010 International Joint Conference on Neural Networks, 2010.