Knowledge-Based Systems 191 (2020) 105257

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Optimizing Deep Belief Echo State Network with a Sensitivity Analysis Input Scaling Auto-Encoder algorithm ☆

Heshan Wang [a], Q.M. Jonathan Wu [b], Jianbin Xin [a,*], Jie Wang [a], Heng Zhang [a]

[a] College of Electrical Engineering, Zhengzhou University, Zhengzhou 450001, PR China
[b] Department of Electrical and Computer Engineering, University of Windsor, Canada, N9B 3P4

ARTICLE INFO

Article history:
Received 10 March 2019
Received in revised form 4 November 2019
Accepted 21 November 2019
Available online 25 November 2019

Keywords:
Echo State Network
Reservoir computing
Recurrent neural network
Sensitivity Analysis Input Scaling
Auto-Encoder
Time-series prediction

ABSTRACT

Echo State Network (ESN) is a specific class of recurrent neural networks, which displays very rich dynamics owing to its reservoir based hidden neurons. ESN has been viewed as a powerful approach to model real-valued time series processes. In order to integrate with deep learning theory, Deep Belief Echo State Network (DBESN) is employed to address the slow convergence in Deep Belief Network (DBN). In DBESN, the DBN part is employed for feature learning in an unsupervised fashion and the ESN part is utilized as a regression layer of DBN. However, the ESN input layer is still not working in an unsupervised status in DBESN. Moreover, ESN's input dimension increases dramatically because of the DBN layer in DBESN. Namely, the DBN layer in DBESN makes the ESN more difficult to construct the input scaling parameters. For purpose of constructing an optimal input weights matrix and input scaling parameters in the ESN layer of DBESN, a novel Sensitivity Analysis Input Scaling Auto-Encoder (SAIS-AE) algorithm is employed in this paper through an unsupervised pre-training process. Initially, the output weights matrix of ESN layer is pre-trained by total input data set. Then, the pre-trained output weights matrix is injected into the input weights matrix of the ESN layer to ensure the specificity of AE. Finally, the input scaling parameters of ESN layer are tuned based on a sensitivity analysis algorithm. Two multivariable sequence tasks and one univariate sequence benchmark are applied to demonstrate the advantage and superiority of SAIS-AE. Extensive experimental results show that our SAIS-AE-DBESN model can effectively improve the performance of DBESN.

1. Introduction

Artificial Neural Networks (ANNs) as one of the most popular black box modeling methods have been widely applied in various applications [1–4]. As a special class of ANN with internal recurrent connections, Recurrent Neural Networks (RNNs) have been paid more attention in recent years for their powerful capacity to model temporal relationships among the time series data set. Owing to their internal memory by dynamic synaptic connections, RNNs are successfully employed to model variety sequence predictions [5,6].

RNNs are natural methods for modeling time series systems with memory. However, the traditional RNNs are hard to achieve long memory spans because of the vanishing gradient problems as described in [7]. To address these vanishing gradient problems, the Long Short-Term Memory (LSTM) networks proposed by Bengio et al. [8] and Echo State Network (ESN) employed by Jaeger et al. [9,10] are currently two choices to solve the vanishing gradient problems. Both LSTM and ESN can learn long-term and short-term dependence information of time series data [11]. The training algorithm of LSTM is usually done by gradient descent using the Back-Propagation Through Time (BPTT) algorithm [12]. However, the training algorithm of ESN is usually done by a least square algorithm [9] which makes the training process simpler than BPTT and guarantees a low computational requirement.

ESN is a new class of efficient RNN based on a randomly generated reservoir and a linear readout mapping from internal layer. ESN is viewed as an effective method to perform on temporal sequences owing to the dynamic internal matrix. The obvious characteristic of ESN is that the reservoir neurons are connected to each other by a pre-determined sparse weights matrix which does not need to be tuned after premier construction. Once the reservoir outputs are an "echo" of the entire input sequences, the ESN can derive an echo state property. Moreover, ESNs have been effectively applied in various sequential areas, such as chaotic system control [13], time-series classification [14,15], time-series

☆ No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.knosys.2019.105257.

* Correspondence to: Kexue Road NO. 100, Zhengzhou 450001, PR China
E-mail addresses: whs7713578@163.com (H. Wang), j.xin@zzu.edu.cn (J. Xin).

https://doi.org/10.1016/j.knosys.2019.105257
0950-7051/

forecasting [16,17], speech recognition [18,19], nonlinear signal processing [20] and batch bioprocesses modeling [21].

Deep Belief Network (DBN) employed by Hinton et al. [22], is a widely studied and generative Deep Neural Network (DNN) for feature extraction. Essentially, the building module of a DBN is a greedy and multi-layer shaping learning model and the learning mechanism is a stack of Restricted Boltzmann Machine (RBM). Unlike other traditional nonlinear models, the obvious merit of DBN is its distinctive unsupervised pre-training to get rid of over-fitting in the training process. In recent years, DBN has drawn increasing attention of community in various application domains such as hyperspectral data classification [23], traffic flow prediction [24] and time series forecasting [25]. In [26], Sun et al. employed a Deep Belief Echo State Network (DBESN) to integrate deep learning theory with ESN learning mechanism. This was the first time that applied the ESN methodology to deep learning. As described in [26], DBESN has more excellent memory capacity and better nonlinear approximation capacity than the conventional ESN. Lots of theoretical conclusions confirm that a deep hierarchical neural network can be more effective than a shallow network [27].

The main advantages of the DBESN architecture compared to the current LSTM based methods can be summarized as follows:

(1) The traditional ESN can learn long-term and short-term dependence information of time series just as LSTM method. Meanwhile, the simple training algorithm of ESN can make the training and testing process simpler than LSTM methods which are usually done by BPTT algorithm.

(2) The DBESN has more excellent memory capacity and better nonlinear approximation capacity than the conventional ESN. In the mean time, a deep hierarchical neural network can be more effective than a shallow network.

Auto-Encoder (AE) is a common unsupervised algorithm to learn efficient data codings in DNN [28,29]. AE tries to transform original input data into output while keeping the deformation amount minimized as far as possible [30]. AE which supplies efficient unsupervised data coding, extracts certain data representation so as to reduce or expand the data dimension. The data dimension can also be kept unaltered by AE. Lately, the AE idea has become more diffusely utilized for learning generative data models [31,32]. Sun et al. presented a sparse AE to learn features from unlabeled measurement data [33]. Wang et al. proposed an AE combined convolution neural network and extreme learning machine for 3D feature learning [34]. Gehring et al. employed a novel de-noising AE to generate bottleneck features from DNN [35]. Totally speaking, most of the AEs are applied to a feed-forward form. On the other hand, the community has made some progress on RNNs based on AE. D'Avino et al. attempted to identify video forgery based on AE and RNN [36]. Pasa and Sperduti proposed a linear AE pre-training method for sequence prediction based on RNN [37]. For ESNs, Chouikhi et al. employed an Echo State Network Auto-Encoder (ESN-AE) technique to extract features from reservoir state based on single and multi layer ESN for sequence classification [38]. In [39], the output weights matrix obtained for ESN-AE is utilized as an initial input weights matrix of ESN, which is then trained on the desired task.

As described in [26], the DBN layer in DBESN is employed for feature learning in an unsupervised fashion and the ESN layer is utilized as a regression layer of DBN. However, the ESN input layer is still not working in an unsupervised status in DBESN. Moreover, ESN's input dimension increases dramatically because of DBN. Namely, the DBN makes the ESN more difficult to construct the input scaling parameters. To address this problem, we borrow the conceptual frameworks proposed in [21,38] to develop an effective pre-training algorithm based on a novel Sensitivity Analysis Input Scaling Auto-Encoder (SAIS-AE) algorithm for time series modeling. Initially, the output weights matrix of ESN layer is pre-trained by the total input data set. Then, the pre-trained output weights matrix is injected into the input weights matrix of the ESN layer to ensure the specificity of AE. Finally, the input scaling parameters of ESN layer are tuned based on a Sensitivity Analysis (SA) algorithm.

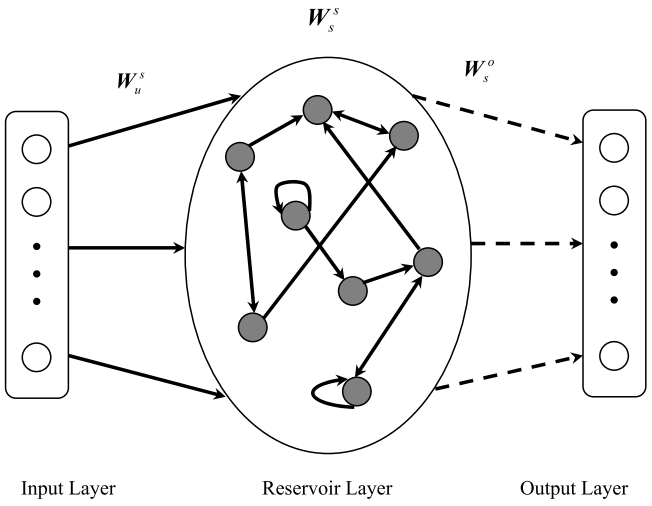

Fig. 1. Basic ESN structure. Black lines are fixed synaptic connections during the training process. Dashed lines represent synaptic connections which need to be trained.

The remainder of this article is partitioned into four sections: In Section 2, a brief overview of projecting and training of DBESN is provided. In Section 3, the detailed steps of SAIS-AE are described. Experimental results are introduced and discussed throughout Section 4. Finally, a brief conclusion is presented in Section 5.

2. Deep belief echo state network

2.1. Echo state network

An ESN is a RNN consisting of K input units with a $K \times N$ input weights matrix, N reservoir neurons with an $N \times N$ sparse weights matrix and L output units with an $L \times N$ output weights matrix. The diagram of ESN is shown in Fig. 1. The reservoir state $\boldsymbol{S}(t)$ at time instant t is the outputs of reservoir layer at that time, $\boldsymbol{S}(t)$ and output $\boldsymbol{O}(t)$ at time step t are given by

$$\boldsymbol{S}(t) = f(\boldsymbol{W}_u^s \cdot \boldsymbol{U}(t) + \boldsymbol{W}_s^s \cdot \boldsymbol{S}(t-1) + \boldsymbol{W}_o^s \boldsymbol{O}^T(t-1)) \quad (1)$$

$$\boldsymbol{O}(t) = f^{\text{out}}(\boldsymbol{S}^T(t) \cdot \boldsymbol{W}_s^o) \quad (2)$$

where f is the reservoir neurons activation function (generally a hyperbolic tangent function), f^{out} is the output layer activation function (typically a linear function), $\boldsymbol{U}(t)$ and $\boldsymbol{O}(t)$ are the input and output signals at time step t respectively, $\boldsymbol{W}_u^s$, $\boldsymbol{W}_s^s$, $\boldsymbol{W}_o^s$ and $\boldsymbol{W}_s^o$ represent the input, reservoir, feedback and readout weights matrix respectively, $\boldsymbol{W}_o^s$ is an $N \times L$ weights matrix if there is close-loop synaptic weights between reservoir and output layers, T denotes the transpose.

$\boldsymbol{W}_s^s$, $\boldsymbol{W}_o^s$ and $\boldsymbol{W}_u^s$ which are randomly generated from a uniform distribution, keep unchanged during training process. Only the readout matrix $\boldsymbol{W}_s^o$ needs to be trained by supervised training algorithm. To account for echo state property, the reservoir connection matrix $\boldsymbol{W}_s^s$ is typically scaled as

$$\boldsymbol{W}_s^s \leftarrow \frac{\alpha}{|\lambda_{\max}|} \boldsymbol{W}_s^s \quad (3)$$

where $|\lambda_{\max}|$ is the spectral radius of $\boldsymbol{W}_s^s$ and $0 < \alpha < 1$ is a scaling parameter.

Data are divided into three categories, namely, training, validation and testing data. Once the training precondition is set, the training can be started through the training data. Afterwards, the reservoir states are collected in a state matrix $\boldsymbol{S}$

$$\boldsymbol{S} = \begin{bmatrix} \boldsymbol{s}^T(1) \\ \boldsymbol{s}^T(2) \\ \vdots \\ \boldsymbol{s}^T(n) \end{bmatrix} \tag{4}$$

and the corresponding target outputs are aggregated in a target matrix $\boldsymbol{O}$

$$\boldsymbol{O} = \begin{bmatrix} \boldsymbol{o}(1) \\ \boldsymbol{o}(2) \\ \vdots \\ \boldsymbol{o}(n) \end{bmatrix} \tag{5}$$

where n is the number of training samples. The readout matrix should then solve a linear regression problem

$$\boldsymbol{S} \cdot \boldsymbol{W}_s^o = \boldsymbol{O} \tag{6}$$

the traditional method is to use the least squares solution

$$\boldsymbol{W}_s^o = \arg\min_w \|\boldsymbol{S}\boldsymbol{w} - \boldsymbol{O}\|^2 \tag{7}$$

where $\|\cdot\|$ denotes the Euclidean norm. The computation of $\boldsymbol{W}_s^o$ is a linear regression, then the readout matrix $\boldsymbol{W}_s^o$ can be done in a single step using the Moore–Penrose pseudo inverse algorithm [9],

$$\boldsymbol{W}_s^o = \tilde{\boldsymbol{S}}\boldsymbol{O} = (\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{S}^T\boldsymbol{O} \tag{8}$$

where $\tilde{\boldsymbol{S}}$ denotes the generalized inverse of $\boldsymbol{S}$, $\boldsymbol{O}$ is target output matrix.

2.2. Diagram of deep belief echo state network

As proposed in [26], the entire network of framework of DBESN consists of two parts: a DBN with multiple RBMs and an ESN regression layer. The DBN is employed for feature learning in an unsupervised fashion and the ESN is utilized as a regression layer of DBN. The diagram of DBESN is shown in Fig. 2.

The output sequence of the last DBN layer can be described as the new input matrix of the ESN layer $\boldsymbol{U}$,

$$\boldsymbol{U} = \begin{bmatrix} \boldsymbol{u}(1) \\ \boldsymbol{u}(2) \\ \vdots \\ \boldsymbol{u}(p) \end{bmatrix} \tag{9}$$

where p denotes the neuron number of the last DBN layer. It can be seen obviously from Fig. 2 that $\boldsymbol{U}$ can be considered as the current state of last RBM in the DBN layer. Since an ESN-based regression layer is proposed after the DBN layer, DBESN can be considered as a complete class of a typical neural network. In this way, the DBESN learning can be divided into two processes, which are the independent DBN learning and the ESN regression learning.

As shown in Fig. 2, a typical RBM consists of two layer units which are binary visible units and hidden units. The RBM commonly utilized as a layer-wise training model in the building of DBN and exhibits a special type of Markov random field with visible units $\boldsymbol{v} = \{0, 1\}^D$ and hidden units $\boldsymbol{h} = \{0, 1\}^F$. A joint configuration of the units, identified by the weighs and biases of RBM, has an energy given by

$$\begin{aligned} E(\boldsymbol{v}, \boldsymbol{h}; \theta) &= -\sum_{i=1}^{D} a_i v_i - \sum_{j=1}^{F} b_j h_j - \sum_{i=1}^{D}\sum_{j=1}^{F} w_{ij} v_i h_j \\ &= -\boldsymbol{a}^T\boldsymbol{v} - \boldsymbol{b}^T\boldsymbol{h} - \boldsymbol{v}^T\boldsymbol{W}\boldsymbol{h} \end{aligned} \tag{10}$$

where $\theta = \{b_i, a_j, w_{ij}\}$, w_{ij} is the weight between visible unit i and hidden unit j; b_i and a_j are bias parameters of visible and hidden unit, respectively. The joint distribution over the visible–hidden units is defined as follows

$$P(\boldsymbol{v}, \boldsymbol{h}; \theta) = \frac{1}{Z(\theta)} exp(-E(\boldsymbol{v}, \boldsymbol{h}; \theta)) \tag{11}$$

where $Z(\theta)$ is a normalizing constant, given by the sum of all possible energy configurations involving the visible and hidden units.

$$Z(\theta) = \sum_{\boldsymbol{v}}\sum_{\boldsymbol{h}} exp(-E(\boldsymbol{v}, \boldsymbol{h}; \theta)) \tag{12}$$

The RBM gives a probability to every input vector via the energy function. In order to weaken the energy as given in Eq. (10), the probability of the training vector can be raised by adjusting θ. The conditional probabilities of hidden unit j and visible unit i are given by logistic functions.

$$p(h_j = 1|\boldsymbol{v}) = \delta(b_j + \sum_i v_i w_{ij}) \tag{13}$$

$$p(h_j = 1|\boldsymbol{h}) = \delta(a_i + \sum_i h_j w_{ij}) \tag{14}$$

$$\delta(x) = \frac{1}{1 + exp(-x)} \tag{15}$$

Once the states of hidden units are given, the input data can be reconstructed by setting each v_i to 1 with the probability of Eq. (14). Then, the states of the hidden units are updated, so that the reconstruction features can be represented by the states of the hidden units.

The learning of w_{ij} is computed by an algorithm which called contrastive divergence (CD) [39]. The update criterion in a weight is given by

$$\Delta w_{ij} = \eta(v_i h_{j\ data} - v_i h_{j\ reconstruction}) \tag{16}$$

where η stands for the learning rate. The suitable value of w_{ij} is obtained through the entire learning procedure. Meanwhile, a trained DBN can be obtained by repeating the above procedure until all the RBMs weights are updated.

3. Optimizing the input weights matrix and input scaling parameters of DBESN with SAIS-AE

3.1. Basic ESN auto-encoder

AE is one of the most common unsupervised tools to automatically learn efficient data features from original unlabeled data. AE aims to make original input unlabeled data almost same as the output data. Generally, there are two parts which are encoding and decoding phases in the AE process. The AE phase which is performed by an unsupervised learning process provides learned features extracted from the training data set.

In [38], an ESN-AE was employed by Chouikhi et al. This was the first time that AE had been utilized in the ESN theory. Basic ESN-AE has three layers, which are input, output and hidden layers as shown in Fig. 3.

The encoding portion is composed of the input weights matrix as well as the reservoir layer. The decoding portion is represented by the readout weights between the reservoir neurons

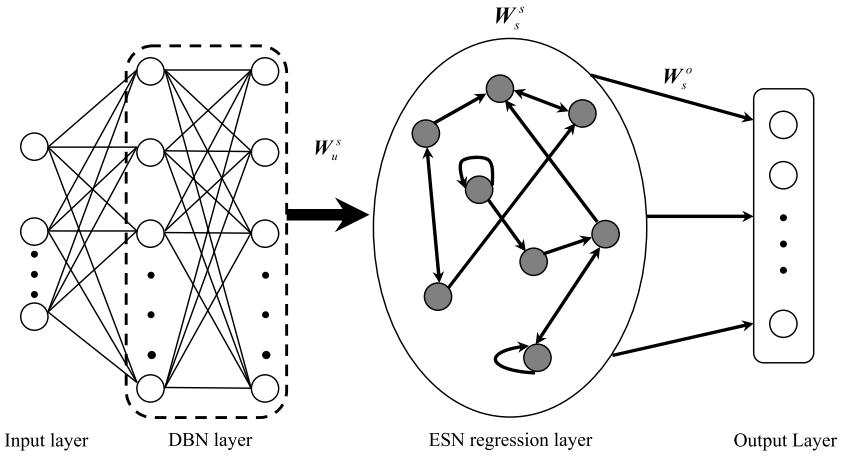

Fig. 2. The architecture of DBESN.

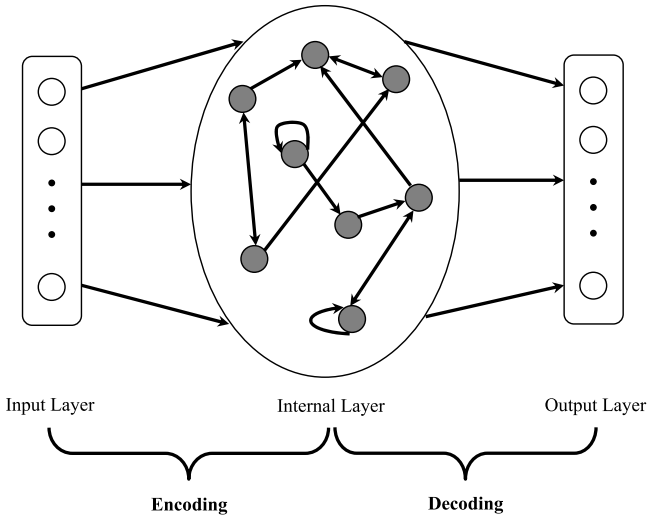

Fig. 3. The basic ESN auto-encoder.

and the readout neurons. Unlike conventional AE based on Back-Propagation (BP) algorithm [40], the training of basic ESN-AE is based on the pseudo-inverse algorithm [9] which is much faster than the BP algorithm. In order to ensure the output equal to its input in AE, the output weights matrix obtained for ESN-AE is then utilized as the initial input weights matrix. ESN-AE is a good starting point to study the weights space of an ESN.

According to the mathematical theory of AE, the outputs of the ESN layer are set to be equal to the inputs of the ESN layer in DBESN. The first step is mapping the inputs to a random feature which is provided by a non-linear transformation. Then, the output weight parameters are solved by the pseudo-inverse algorithm. Therefore, the new input weights matrix can be obtained from basic ESN-AE which are described in Eqs. (17) and (18),

$$\boldsymbol{W}_s^o = \tilde{\boldsymbol{S}}\boldsymbol{O} = (\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{S}^T\boldsymbol{U} \tag{17}$$

$$\boldsymbol{W}_u^s = (\boldsymbol{W}_s^o)^T \tag{18}$$

where the ESN input signals are collected in an input training matrix $\boldsymbol{U}$ as described in Eq. (9). Thus, the output of the ESN layer is equal to the input of the ESN layer as we aim to approximate the inputs.

3.2. Sensitivity analysis input scaling auto-encoder algorithm

In [21], Wang et al. provided a SAIS algorithm which an ESN without tuning is established firstly and then its input scaling parameters are tuned based on SA [41]. As described in [21], SAIS is an effective algorithm to design the input scaling parameters of ESN with more than one input sequence.

SA is the study of how uncertainty in the output of a model can be attributed to different sources of uncertainty in the model input factors. SA algorithm based on variance decomposition is a global method and can dispose of nonlinear models. Considering the output of a nonlinear model

$$\boldsymbol{O} = f(\boldsymbol{U}) = f(\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_p) \tag{19}$$

where $\boldsymbol{O}$ is the output and $\boldsymbol{U} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_p)$ contains p independent input factors, each one varying over its own probability density function. The first-order sensitivity index proposed in [41] can be decomposed the model function f into summands of increasing dimensionality.

$$f(\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_p) = f_0 + \sum_{i=1}^{p} f_i(\boldsymbol{u}_i) + \sum_{i=1}^{p}\sum_{j=i+1}^{p} f_{ij}(\boldsymbol{u}_i, \boldsymbol{u}_j) + \cdots + f_{1,\ldots,p}(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_p) \tag{20}$$

where f_0 is a constant equal to the expectation of the model output, and the integrals of every summand over any of its own variables are zero as described in Eq. (21)

$$\int_0^1 f_{is}(u_{i1}, u_{i2}, \ldots, u_{ip})du_{ik} = 0 \tag{21}$$

where $1 < k < p$. Consequently, all the summands are mutually orthogonal. The total unconditional variance can be defined as

$$V(\boldsymbol{O}) = \int_{\Omega^p} f^2(\boldsymbol{u})d\boldsymbol{u} - f_0^2 \tag{22}$$

where Ω^p representing the p dimensional unit hyperspace (i.e., parameter ranges are scaled between 0 and 1). The partial variances are computed from each of the terms in Eq. (20).

$$V_{i1,\ldots,is} = \int_0^1 \cdots \int_0^1 f_{i1,\ldots,is}^2(u_{i1}, \ldots, u_{is})du_{i1}\ldots du_{is} \tag{23}$$

Based on the assumption that the parameters are mutually orthogonal, the variance decomposition can be obtained as

$$V(\boldsymbol{O}) = \sum_{i=1}^{p} V_i + \sum_{i=1}^{p-1}\sum_{j=i+1}^{p} V_{ij} + \cdots V_{1,\ldots,p} \tag{24}$$

Consequently, the variance contributions to the total output variance of individual parameters and parameter interactions can be determined. These contributions are characterized by the ratio of the partial variance to the total variance which can be called the Sobol' Sensitivity Indices (SI) [41].

$$\text{First-order SI: } S_i = \frac{V_i}{V} \tag{25}$$

$$\text{Second-order SI: } S_{ij} = \frac{V_{ij}}{V} \tag{26}$$

$$\text{Total SI: } S_{Ti} = S_i + \sum_{j \neq i} S_{ij} + \cdots = 1 - \frac{V_{\sim i}}{V} \tag{27}$$

where $V_{\sim i}$ is the variation of all parameters except u_i. For additive models and under the assumption that orthogonal input factors, S_{Ti} and S_i are equal, the sum of all S_i is 1. For non-additive models, S_{Ti} is greater than S_i and the sum of all S_i is less than 1. Meanwhile, the sum of all S_{Ti} is greater than 1.

The idea presented in this paper is as follows. Firstly, the output sequences of the last DBN layer which can be described as the new input matrix of ESN layer are collected. The output weights matrix of ESN layer is pre-trained by the total input data set according to Eq. (17). Then, the pre-trained output weights matrix is injected into the input weights matrix of the ESN layer to ensure the specificity of AE according to Eq. (18). Finally, the new input scaling parameters of ESN layer are tuned based on SA.

The ESN layer can be trained by minimizing a given loss function. In most cases, the model performance is evaluated via Normalized Mean Square Error (*NMSE*),

$$NMSE = \sum_{t=1}^{N} \frac{(\boldsymbol{o}(t) - \hat{\boldsymbol{o}}(t))^2}{N\sigma^2} \tag{28}$$

where $\hat{\boldsymbol{o}}(t)$ is the actual readout output, $\boldsymbol{o}(t)$ is the desired output, σ^2 denotes the variance of $\boldsymbol{o}(t)$, and N is the total number of $\boldsymbol{o}(t)$.

The main steps of SAIS-AE are conducted as follows,

1. All the sequences are divided into three classes, training set, validation set and testing set.

2. A DBESN with the default input scaling parameters $\boldsymbol{\beta}$ which is a unit vector is established.

$$\boldsymbol{\beta} = [\beta_1, \beta_2, \ldots, \beta_p] = [1, 1, \ldots, 1] \tag{29}$$

The initial input weights matrix $\boldsymbol{W}_u^s$ is generated randomly from a uniform distribution over an interval $[-1, 1]$. The reservoir neurons are standard sigmoid units with a $f = tanh$ activation function. All the other parameters in the established DBESN such as the number of DBN layers, the neuron number of each RBM, are chosen on a validation set until the established DBESN achieve satisfactory performance.

3. The output sequences of the last DBN layer are collected in the independent DBN learning process. Then, the output sequences of the last DBN layer can be described as the new input matrix of ESN layer according to Eq. (9).

4. The reservoir state $\boldsymbol{S}$ of training data is collected through Eq. (4). The established DBESN starts at time $t = 1$ with zero starting state $\boldsymbol{s}(t) = 0$. During the collecting period, the initial washout data is discarded while the remaining reservoir states $\boldsymbol{s}(t)$ is sampled into the state matrix $\boldsymbol{S}$. Meanwhile, the pre-trained output weights matrix of ESN layer is then calculated and saved by Eq. (17).

5. According to the basic ESN-AE mechanism, the outputs of the ESN layer are set to be equal to the inputs of the ESN layer in DBESN. The optimal input weights matrix of ESN layer in the DBESN model can be gained by Eq. (18).

6. The new DBESN with the optimal input weights matrix obtained in step 5 is reestablished.

7. The total SI $\boldsymbol{S}_T = [S_{T1} \quad S_{T2} \quad \cdots \quad S_{Tp}]$ of each input variable (output variable of the last DBN layer) is calculated using Sobol' quasi-random sequences [41] according to Eq. (27). The inputs are then sorted by decreasing sensitivity numbers for efficient tuning of the input scaling parameters.

8. The input scaling parameters $\boldsymbol{\beta}$ is adjusted based on $\boldsymbol{S}_T$. SA can determine the most influential inputs of a model. The input scaling parameters of the important input variable should be increased and vice versa. In this study, we increase the scaling parameter β_m to 2 of the most significant input. Another input scaling parameter β_i can be adjusted proportionally by $\beta_i/\beta_m = S_{Ti}/S_{Tm}$. Thus, the new inputs scaling parameter vector $\hat{\boldsymbol{\beta}}$ is then obtained.

9. The final DBESN model with the optimal input scaling parameter $\hat{\boldsymbol{\beta}}$ and the optimal input weights matrix obtained in step 5 is established.

10. The new DBESN is trained and tested by traditional pseudo-inverse algorithm as proposed in Eq. (8). The final performance $NMSE_{test}$ of the new DBESN is calculated through the testing data.

The flow diagram of DBESN designed by SAIS-AE is shown in Fig. 4.

4. Experimental results

A comprehensive experimental evaluation for our DBESN model based on SAIS-AE is provided in the following section, considering three time-series benchmarks which are widely used in the ESN literatures [16,26,42–44]: Nonlinear Autoregressive Moving Average (NARMA) system, Heat exchangers (HX) system and Mackey–Glass (MG) time series system. In the following experiments, each dataset is divided into three parts for training, validating, and testing, respectively. The validating data is used to generate the parameters in the DBESN model such as the number of DBN layers, the neuron number of each RBM until the established DBESN achieve satisfactory performance. The percentage improvement in performance of the test model is defined as,

$$IM\% = \frac{NMSE_{ESN} - NMSE_{test}}{NMSE_{ESN}} \times 100\% \tag{30}$$

where $NMSE_{ESN}$ is the performance of ESN and $NMSE_{test}$ is the performance of the test model. We can justify the superiority of DBESN, AE-DBESN and SAIS-AE-DBESN algorithms through *IM%*.

Our SAIS-AE-DBESN model is compared with those of classical ESN, DBESN, and AE-DBESN models. In order to ensure the fairness of comparison, all the ESN layer parameters in DBESN and classical ESN such as reservoir neurons and weights, initial input weights matrix are the same value. The entire spectral radius used in classical ESN and DBESN of our experiments is 0.95. Detailed parameter settings of the evaluated models can be found in Table 1

4.1. NARMA system

As a multivariable time series system, the current output of this task not only depends on the previous input but also the previous output. The task is described by the following equation.

$$o(t+1) = 0.3o(t) + 0.05o(t)\sum_{i=0}^{9} o(t-i) + 1.5u(t-9)u(t) + 0.1 \tag{31}$$

where $o(t)$ is the system output at time t, $u(t)$ is the system input at time t (i.i.d stream of values generated uniformly from [0, 0.5]). In general, modeling this system is difficult due to the non-linearity and possibly long memory. A length of 10,000 items of

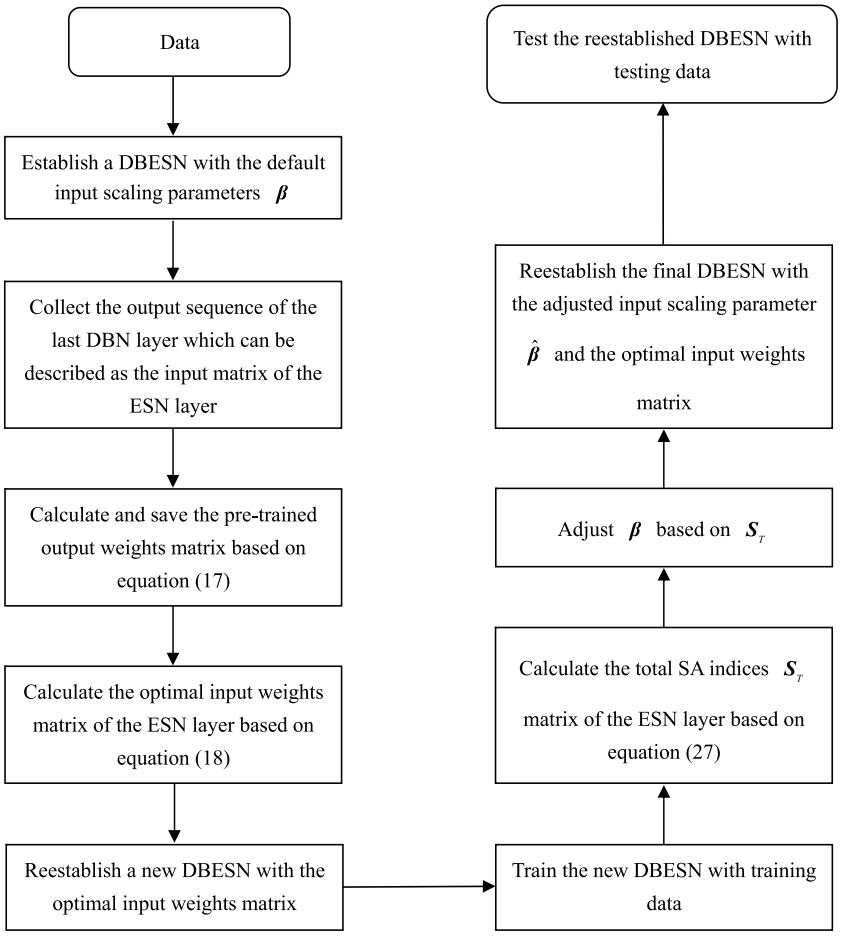

Fig. 4. The flow diagram of DBESN designed by SAIS-AE algorithm.

Table 1
Parameter values for the evaluated models of different tasks.

Task	Batch-size	DBN layers	Units in each RBM	f^{out}	N	Teacher scaling	Learning rate	Feedback scaling
NARMA	100	2	10	Linear	500	1	0.01	0
HX	100	2	10	Linear	500	1	0.01	0
MG	100	1	10	tanh	500	0.8	0.05	1

Table 2
The testing *NMSE* performance of SAIS-AE-DBESN, ESN, DBESN and AE-DBESN for the NARMA task.

Method	Testing *NMSE*	*IM*%
ESN	0.04790	–
DBESN	0.00994	79.25%
AE-DBESN	0.00883	81.57%
SAIS-AE-DBESN	**0.00721**	**84.95%**

NARMA sequence is generated in this paper. The first 4000 items are training data, the next 4000 items are utilized as validation data and the last 2000 items are used for testing the performance. The initial washout period which exists in the training and testing process is 200 in this experiment.

The prediction results over a selected length from time step 8201–8300 obtained by SAIS-AE-DBESN and ESN for 10th order NARMA task are shown in Fig. 5. The *NMSE* performance of SAIS-AE-DBESN, ESN, DBESN and AE-DBESN for 10th order NARMA task is shown in Table 2. The new input scaling parameters vector of SAIS-AE-DBESN obtained by Eq. (27) is $\hat{\boldsymbol{\beta}} = [0.35, 0.13, 2, 0.96, 1.20, 0.38, 0.52, 1.23, 0.48, 0.63]$.

4.2. Heat exchangers system

The HX system introduced in [43] is a multivariable complex nonlinear benchmark which is an effective heat transfer between the two fluids by virtue of their temperature differences. The input and output sequences of the HX system are shown in Fig. 6. The difficulties which are complicated heat, fluid flow geometries and turbulence in the flow etc make the HX modeling peculiarly hard.

The total length of the HX system is 4000. The first 2000 items are training data, the next 1000 items are utilized as validation data and the last 1000 items are used for testing the performance. The initial washout period which exists in the training and testing process is 200 in this experiment.

The prediction results over a selected length from time step 3201–4000 obtained by SAIS-AE-DBESN and ESN for the HX task are shown in Fig. 7. The *NMSE* performance of SAIS-AE-DBESN, ESN, DBESN and AE-DBESN for the HX task is shown in Table 3. The new input scaling parameters vector of SAIS-AE-DBESN obtained by Eq. (27) is $\hat{\boldsymbol{\beta}} = [0.28, 0.57, 0.33, -0.93, 0.16, 0.58, 2, -0.33, -1.22, -1.24]$.

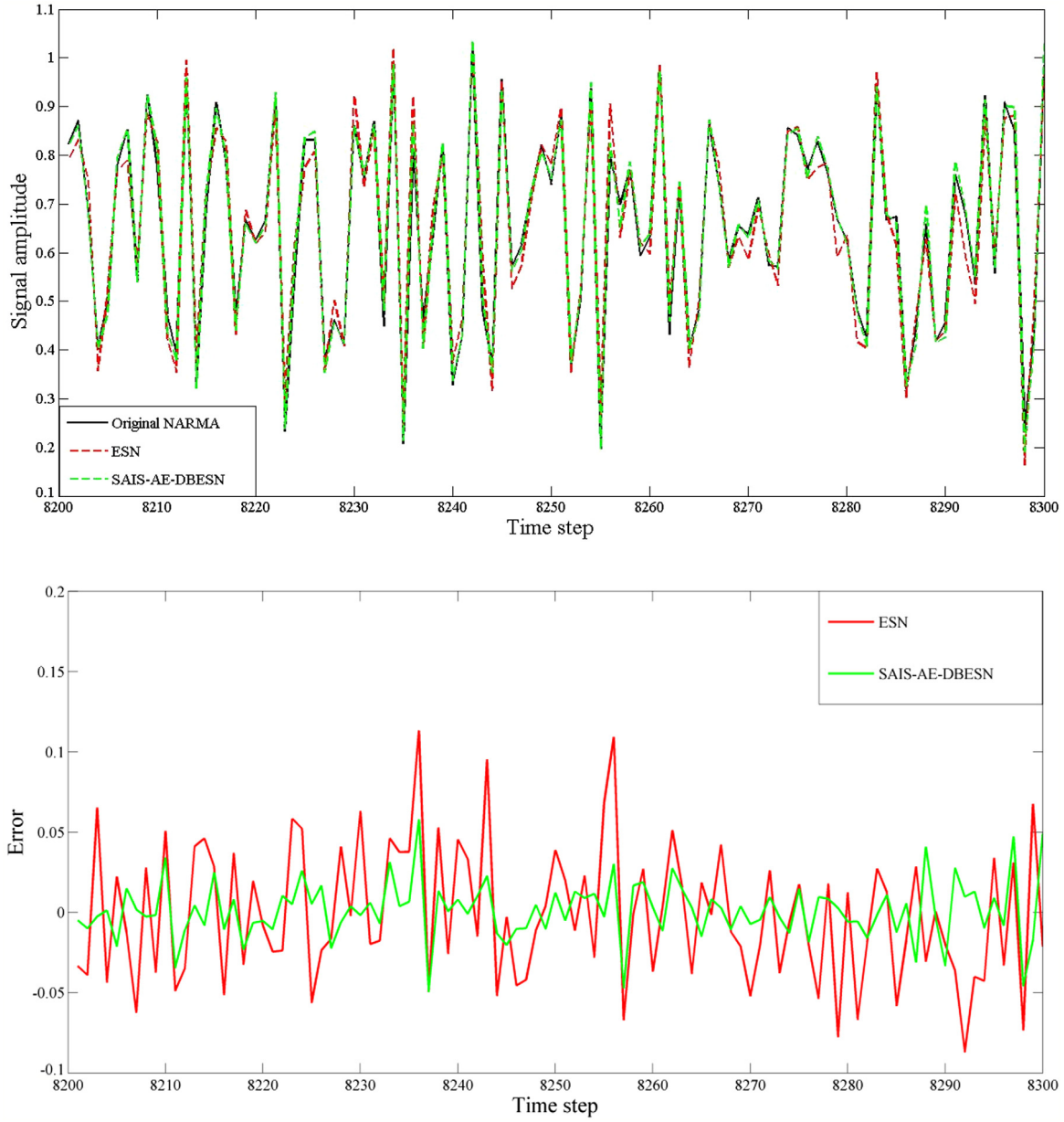

Fig. 5. Output and error performance of SAIS-AE-DBESN and ESN for the NARMA task.

Table 3
The testing *NMSE* performance of SAIS-AE-DBESN, ESN, DBESN and AE-DBESN for the HX task.

Method	Testing *NMSE*	*IM%*
SN	0.3217	–
DBESN	0.3160	1.77%
AE-DBESN	0.1683	47.68%
SAIS-AE-DBESN	**0.0969**	**69.88%**

4.3. Mackey–Glass sequence

The MG sequence system [45] is a univariate benchmark with a chaotic attractor for the sequence prediction model. MG sequence system has been successfully applied to ESN models in various kinds of literatures with a good performance. The MG time series is derived from a time-delay differential mechanism with the following equation.

$$\frac{\partial o(t)}{\partial t} = \frac{0.2o(t-\alpha)}{1+o(t-\alpha)^{10}} - 0.1o(t) \tag{32}$$

The most used values for α are 17 and 30. The chaotic behavior can be an exhibit in the MG system if the delay time $\alpha > 16.8$. Jaeger [9] employed an 84 multi-step prediction model utilizing ESN with 1000 reservoir neurons and a synaptic feedback connection for the MG system.

In our experiment, 10,000 values are generated from Eq. (32) with $\alpha = 17$. The whole sequence used in this paper is divided into three parts: The first 3000 items are training data, the next 2000 items are utilized as validation data and the last 4200 items are utilized for the 84-step iterative prediction of 50 times. The initial washout period which exists in the training and testing process is 1000 in this experiment. The normalized root mean

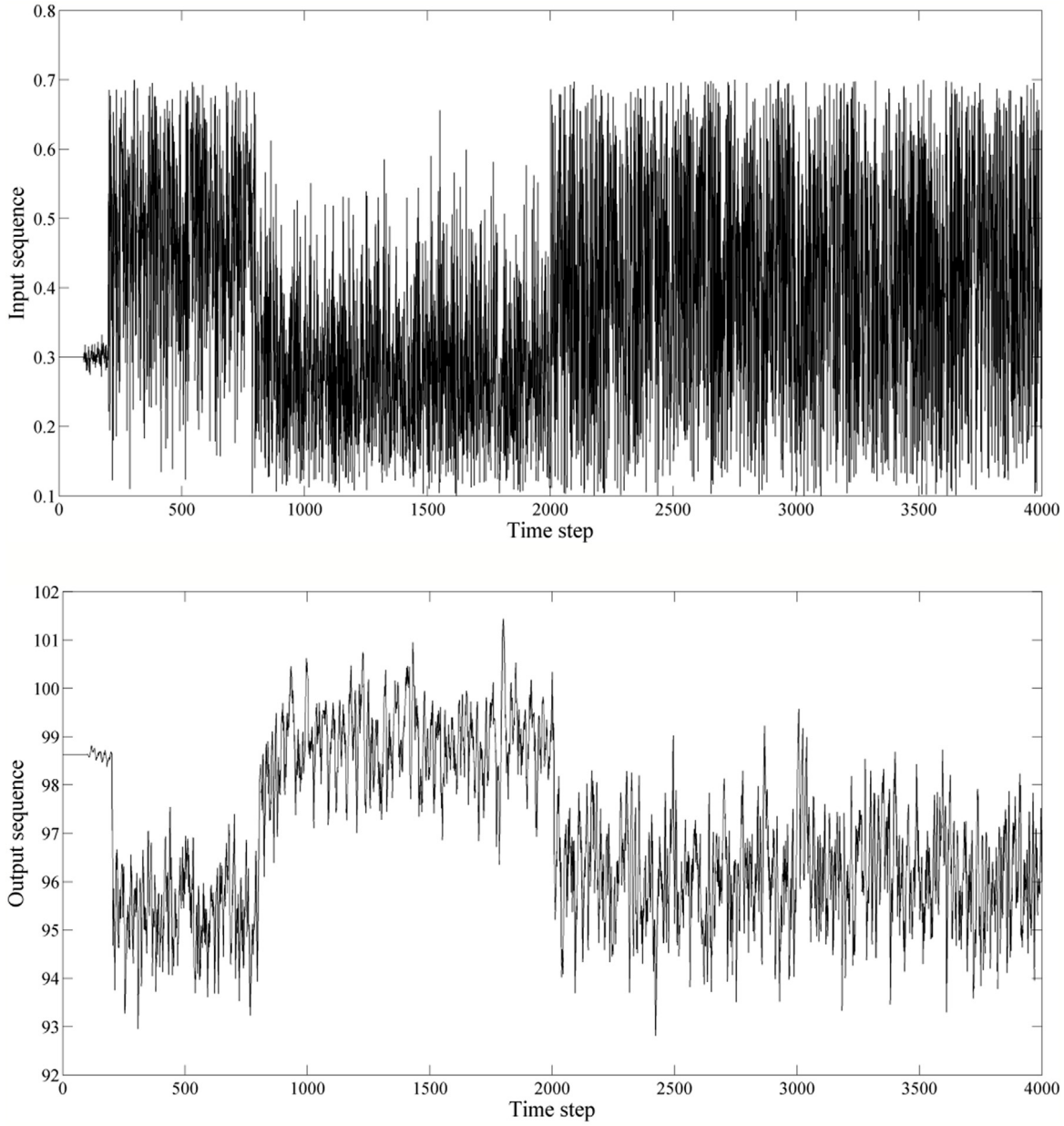

Fig. 6. Sequences of the HX system.

square error at the 84th time step $NRMSE_{84}$ used here is

$$NRMSE_{84} = \sqrt{\sum_{j=1}^{N} \frac{(\hat{o}_i(84) - o_i(84))^2}{N\sigma^2}} \tag{33}$$

where $\hat{o}_i(84)$ and $o_i(84)$ are the predicted output and desire output at time step 84. A constant 0.02 is used as a bias input. A feedback synaptic connection is required in this multi-step prediction model as described before.

The new input scaling parameters of SAIS-AE-DBESN is the same as AE-DBESN because the input of MG task is a constant, so SAIS-AE-DBESN presents the same performance as AE-DBESN in this experiment. The 84-step prediction error performance over a selected length from time step 6001–6084 utilizing SAIS-AE-DBESN and ESN for MG task is shown in Fig. 8. The output performance for 84-step prediction MG system is not shown here because the error is too small to be seen in the figure. The $NRMSE_{84}$ performance of SAIS-AE-DBESN, ESN, DBESN and AE-DBESN for MG sequence task is shown in Table 4.

Table 4
The testing $NRMSE_{84}$ performance of SAIS-AE-DBESN, ESN, DBESN and AE-DBESN for the MG task.

Method	Testing $NRMSE_{84}$	*IM%*
ESN	0.00427	–
DBESN	9.038e−4	78.83%
AE-DBESN	**9.716e−5**	**97.72%**
SAIS-AE-DBESN	**9.716e−5**	**97.72%**

4.4. Comparison of SAIS-AE-DBESN and various other time series models

In order to evaluate the performance of the SAIS-AE-DBESN model, several other models, such as traditional Elman Neural Network (Elman) [46], Sensitive Iterative Pruning Algorithm Echo State Network [16] (SIPA-ESN), Binary Particle Swarm Optimized Echo State Network [42] (BPSO-ESN), are implemented for the

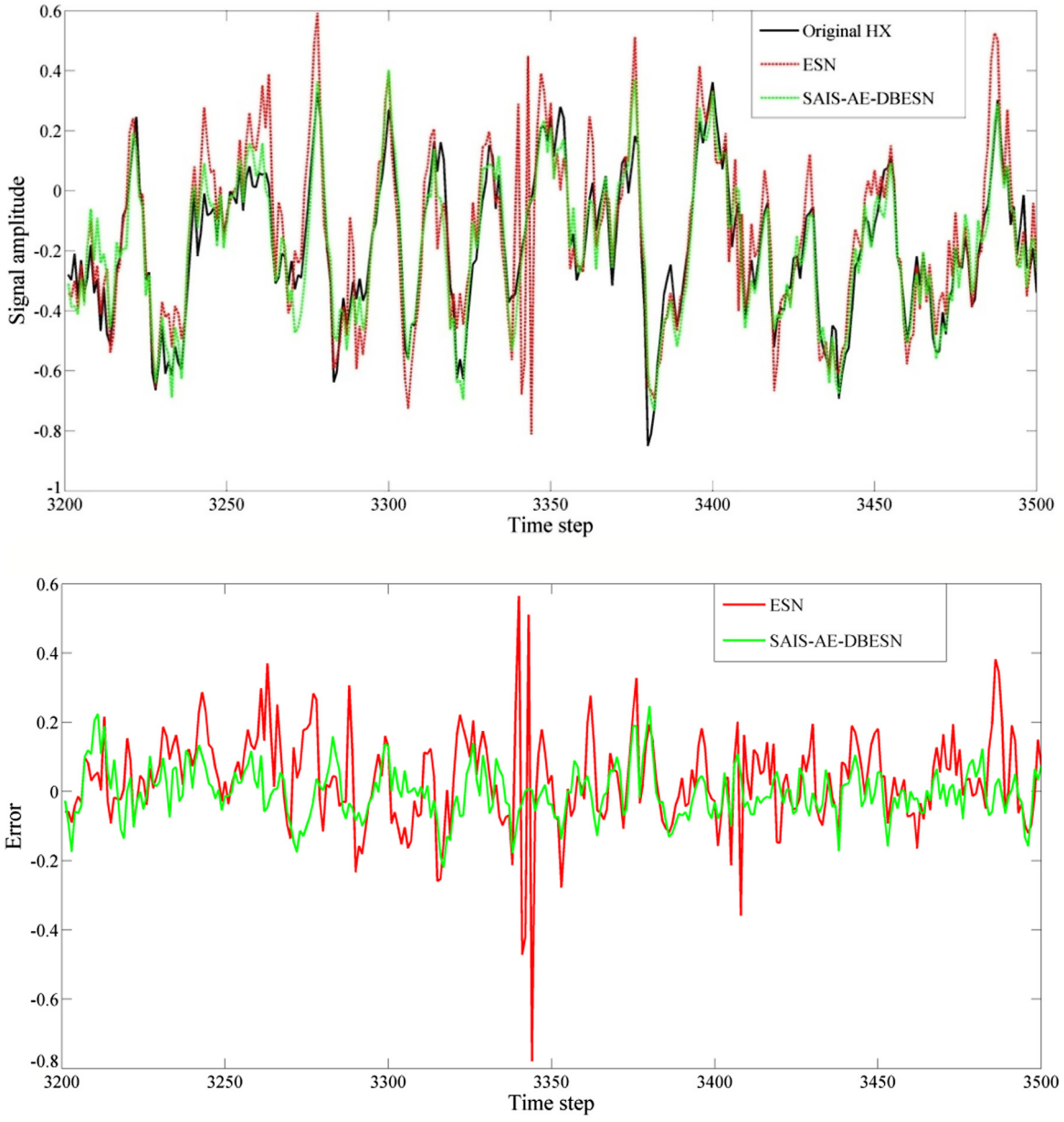

Fig. 7. Output and error performance of SAIS-AE-DBESN and ESN for the HX task.

comparison. The comparison of SAIS-AE-DBESN and various other models is presented in Table 5.

For verifying the robustness of the proposed method, an independent white Gaussian noise sequence is added to the three time-series benchmarks. The testing performance of SAIS-AE-DBESN and ESN with white Gaussian noise for NARMA, HX and MG tasks is presented in Table 6.

4.5. Discussion

The experimental results clearly show that SAIS-AE allows significantly improve the performance of DBESN according to Figs. 5, 7 and 8. The results show that AE is an efficient algorithm to select a better input weights matrix than a randomly generated input weights matrix in the ESN layer of DBESN. According to Tables 2 and 3, the SAIS-AE-DBESN model significantly outperforms AE-DBESN without input scaling tuning process. This shows that the input scaling parameters have a great influence on the output performance in the case of multivariable ESN models. Meanwhile, SAIS is an efficient algorithm to design better input scaling parameters in DBESN. According to Table 6, the experimental results show that the proposed SAIS-AE-DBESN model is accurate and robust for noisy nonlinear time series prediction.

5. Conclusion

A SAIS-AE-DBESN model is proposed in this paper to optimize the input weights matrix and input scaling parameters in the ESN layer of DBESN through an unsupervised pre-training AE and a SA process. Two multivariable sequence tasks and one univariate sequence task are applied to demonstrate the advantage and superiority of SAIS-AE. Extensive experimental results show that our SAIS-AE-DBESN model can effectively improve the performance of DBESN. In the original DBESN model, although the DBN layer can work in an unsupervised pattern, however the ESN input layer is not working in an unsupervised status. Our SAIS-AE method can not only make the ESN layer of DBESN work in an unsupervised fashion but also optimize the input scaling parameters greatly.

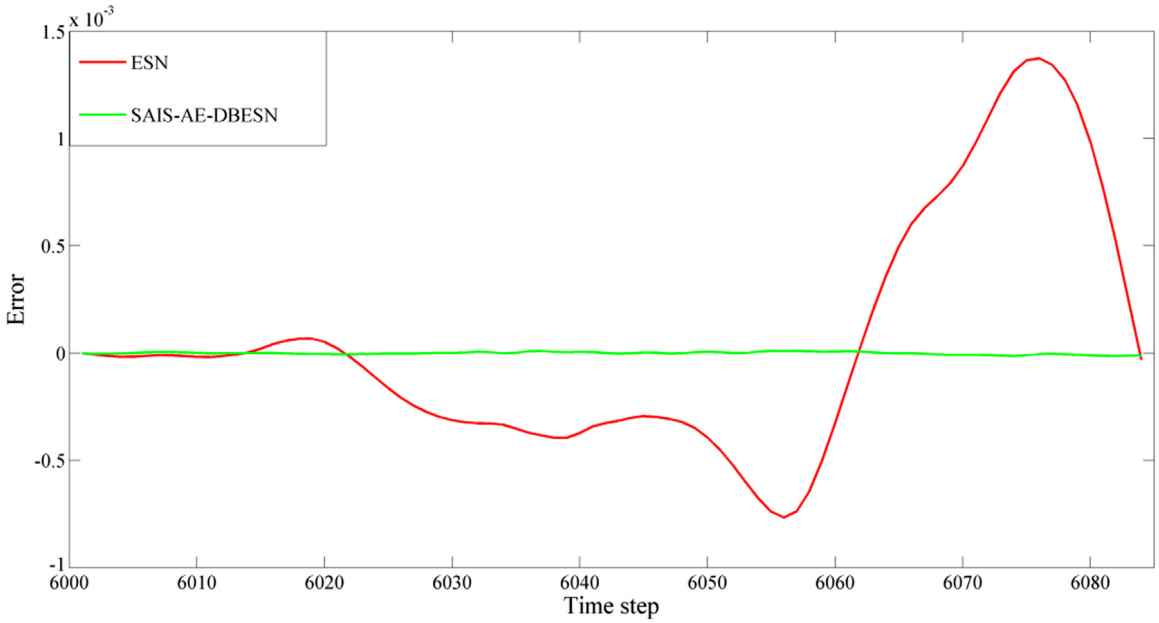

Fig. 8. Error performance of SAIS-AE-DBESN and ESN for the MG task.

Table 5
The comparison of SAIS-AE-DBESN and various other models.

Task	Elman (N = 10)	SIPA-ESN (N = 500)	BPSO-ESN (N = 700)	SAIS-AE-DBESN (N = 500)
NARMA	$NMSE = 0.7814$	$NMSE = 0.0177$	$NMSE = 0.0201$	$NMSE =$ **0.0043**
HX	$NMSE = 1.0372$	$NMSE = 0.1033$	$NMSE = 0.2271$	$NMSE =$ **0.0969**
MG	$NRMSE_{84} = 0.7871$	$NRMSE_{84} = 5.982e{-4}$	$NRMSE_{84} = 2.592e{-4}$	$NRMSE_{84} =$ **9.716e−5**

Table 6
The testing performance of SAIS-AE-DBESN and ESN with white Gaussian noise for NARMA, HX and MG tasks.

Task	ESN	SAIS-AE-DBESN
NARMA	$NMSE = 0.0511$	$NMSE =$ **0.0105**
HX	$NMSE = 0.4307$	$NMSE =$ **0.1024**
MG	$NRMSE_{84} = 0.0328$	$NRMSE_{84} =$ **8.43e−3**

Acknowledgments

The authors gratefully acknowledge the support of the following foundations: National Natural Science Foundation of China (61603343) and National Natural Science Foundation of China (61703372). The authors gratefully acknowledge financial support from China Scholarship Council.

References

[1] Z. Yang, J. Peng, Y. Liu, Adaptive neural network force tracking impedance control for uncertain robotic manipulator based on nonlinear velocity observer, Neurocomputing 331 (2019) 263–280.
[2] J. Peng, Z. Yang, T. Ma, Position/force tracking impedance control for robotic systems with uncertainties based on adaptive Jacobian and neural network, Complexity (2019) 1406534.
[3] S. Poria, E. Cambria, A. Gelbukh, Aspect extraction for opinion mining with a deep convolutional neural network, Knowl.-Based Syst. 108 (2016) 42–49.
[4] Y. Ma, P. Niu, X. Zhang, G. Li, Research and application of quantum-inspired double parallel feed-forward neural network, Knowl.-Based Syst. 136 (2017) 140–149.
[5] S. Benabderrahmane, N. Mellouli, M. Lamolle, On the predictive analysis of behavioral massive job data using embedded clustering and deep recurrent neural networks, Knowl.-Based Syst. 151 (2018) 95–113.
[6] R. Chandra, Competition and collaboration in cooperative coevolution of Elman recurrent neural networks for time-series prediction, IEEE Trans. Neural Netw. Learn. Syst. 26 (2015) 3123–3136.
[7] Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE Trans. Neural Netw. 5 (1994) 157–166.
[8] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (1997) 1735–1780.
[9] H. Jaeger, H. Haas, Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication, Science 304 (2004) 78–80.
[10] H. Jaeger, The "Echo State" Approach to Analysing and Training Recurrent Neural Networks, Technology GMD Technical Report 148, German National Research Center for Information, Germany, 2001.
[11] H. Jaeger, Long Short-Term Memory in Echo State Networks: Details of a Simulation Study, Jacobs University Bremen, 2012.
[12] P.J. Werbos, Backpropagation through time: what it does and how to do it, Proc. IEEE 78 (1990) 1550–1560.
[13] G. Li, P. Niu, W. Zhang, Y. Zhang, Control of discrete chaotic systems based on echo state network modeling with an adaptive noise canceler, Knowl.-Based Syst. 35 (2012) 35–40.
[14] L. Wang, Z. Wang, S. Liu, An effective multivariate time series classification approach using echo state network and aptive differential evolution algorithm, Expert Syst. Appl. 43 (2016) 237–249.
[15] Q. Ma, L. Shen, W. Chen, J. Wang, J. Wei, Z. Yu, Functional echo state network for time series classification, Inform. Sci. 373 (2016) 1–20.
[16] H. Wang, X. Yan, Improved simple deterministically constructed cycle reservoir network with sensitive iterative pruning algorithm, Neurocomputing 145 (2014) 353–362.
[17] Z.W. Shi, M. Han, Support vector echo-state machine for chaotic time-series prediction, IEEE Trans. Neural Netw. 18 (2007) 359–372.
[18] E. Trentin, S. Scherer, F. Schwenker, Emotion recognition from speech signals via a probabilistic echo-state network, Pattern Recognit. Lett. 66 (2015) 4–12.
[19] M.D. Skowronski, J.G. Harris, Automatic speech recognition using a predictive echo state network classifier, Neural Netw. 20 (3) (2007) 414–423.
[20] Y. Xia, B. Jelfs, M.M. Van Hulle, J.C. Principe, D.P. Mandic, An augmented echo state network for nonlinear adaptive filtering of complex noncircular signals, IEEE Trans. Neural Netw. 22 (1) (2011) 74–83.
[21] H. Wang, X. Yan, Reservoir computing with sensitivity analysis input scaling regulation and redundant unit pruning for modeling fed-batch bioprocesses, Ind. Eng. Chem. Res. 53 (16) (2014) 6789–6797.
[22] G. Hinton, R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.

[23] Y. Chen, X. Zhao, X. Jia, Spectral–spatial classification of hyperspectral data based on deep belief network, IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 8 (2015) 2381–2392.
[24] L. Zhao, Y. Zhou, H. Lu, H. Fujita, Parallel computing method of deep belief networks and its application to traffic flow prediction, Knowl.-Based Syst. 163 (2019) 972–987.
[25] M. Qin, Z. Li, Z. Du, Red tide time series forecasting by combining ARIMA and deep belief network, Knowl.-Based Syst. 125 (2017) 39–52.
[26] X. Sun, T. Li, Q. Li, Y. Huang, Y. Li, Deep belief echo-state network and its application to time series prediction, Knowl.-Based Syst. 130 (2017) 17–29.
[27] L. Deng, J. Chen, Sequence classification using the high-level features extracted from deep neural networks, in: IEEE International Conference on Acoustics, Speech and Signal Processing, 2014, pp. 6844–6848.
[28] W. Wu, J. Zhao, C. Zhang, et al., Improving performance of tensor-based context-aware recommenders using bias tensor factorization with context feature auto-encoding, Knowl.-Based Syst. 128 (2017) 71–77.
[29] S. Haidong, J. Hongkai, L. Xingqiu, et al., Intelligent fault diagnosis of rolling bearing using deep wavelet auto-encoder with extreme learning machine, Knowl.-Based Syst. 140 (2018) 1–14.
[30] M. Längkvist, A. Loutfi, Learning feature representations with a cost-relevant sparse autoencoder, Int. J. Neural Syst. 25 (1) (2015) 1450034.
[31] Y. Bengio, L. Yao, G. Alain, P. Vincent, Generalized denoising auto-encoders as generative models, in: Neural Information Processing Systems Conference, 2013, pp. 899–907.
[32] D.P. Kingma, S. Mohamed, D.J. Rezende, M. Welling, Semi-supervised learning with deep generative models, in: Neural Information Processing Systems Conference, 2014, pp. 3581–3589.
[33] W. Sun, S. Shao, R. Zhao, R. Yan, X. Zhang, X, Chen, A sparse auto-encoder-based deep neural network approach for induction motor faults classification, Measurement 89 (2016) 171–178.
[34] Y. Wang, Z. Xie, K. Xu, Y. Dou, Y. Lei, An efficient and effective convolutional auto-encoder extreme learning machine network for 3d feature learning, Neurocomputing 174 (2016) 988–998.
[35] J. Gehring, Y. Miao, F. Metze, A. Waibel, Extracting deep bottleneck features using stacked auto-encoders, in: IEEE International Conference on Speech and Signal Processing, IEEE, 2013, pp. 3377–3381.
[36] D. D'Avino, D. Cozzolino, G. Poggi, L. Verdoliva, Autoencoder with recurrent neural networks for video forgery detection, Electron. Imaging (7) (2017) 92–99.
[37] L. Pasa, A. Sperduti, Pre-training of recurrent neural networks via linear autoencoders, in: Neural Information Processing Systems Conference, 2014, pp. 3572–3580.
[38] N. Chouikhi, B. Ammar, A.M. Alimi, Genesis of basic and multi-layer echo state network recurrent autoencoders for efficient data representations, 2018, arXiv preprint, arXiv:180408996.
[39] G.E. Hinton, Training products of experts by minimizing contrastive divergence, Neural Comput. 14 (8) (2002) 1711–1800.
[40] R. Hecht-Nielsen, Theory of the backpropagation neural network, in: Neural Networks for Perception, Academic Press, 1992, pp. 65–93.
[41] I.M. Sobol', On sensitivity estimation for nonlinear mathematical models, Mat. Model. 2 (1990) 112–118.
[42] H. Wang, X. Yan, Optimizing the echo state network with a binary particle swarm optimization algorithm, Knowl.-Based Syst. 86 (2015) 182–193.
[43] S. Bittanti, L. Piroddi, Nonlinear identification and control of a heat exchanger: a neural network approach, J. Franklin Inst. B 334 (1) (1997) 135–153.
[44] G. Dıaz, M. Sen, K.T. Yang, et al., Dynamic prediction and control of heat exchangers using artificial neural networks, Int. J. Heat Mass Transfer 44 (9) (2001) 1671–1679.
[45] M.C. Mackey, L. Glass, Oscillation and chaos in physiological control systems, Science 197 (1977) 287–289.
[46] J.L. Elman, Finding structure in time, Cognitive science 14 (1990) 179–211.