Deep belief echo-state network and its application to time series prediction

Xiaochuan Sun\textsuperscript{a,b}, Tao Li\textsuperscript{b}, Qun Li\textsuperscript{b}, Yue Huang\textsuperscript{b}, Yingqi Li\textsuperscript{a}

\textsuperscript{a} School of Information Engineering, North China University of Science and Technology, Tangshan, 063009, PR China
\textsuperscript{b} School of Computer Science and Technology, Nanjing University of Posts and Telecommunications, Nanjing, 210023, PR China

\textbf{ARTICLE INFO}

Article history:
Received 19 November 2016
Revised 19 May 2017
Accepted 23 May 2017
Available online 25 May 2017

Keywords:
Deep belief network
Echo state network
Memory capacity
Time series prediction

\textbf{ABSTRACT}

Deep belief network (DBN) has attracted many attentions in time series prediction. However, the DBN-based methods fail to provide favorable prediction results due to the congenital defects of the back-propagation method, such as slow convergence and local optimum. To address the problems, we propose a deep belief echo-state network (DBEN) for time series prediction. In the new architecture, DBN is employed for feature learning in an unsupervised fashion, which can effectively extract hierarchical data features. An innovative regression layer, embedding an echo-state learning mechanism instead of the traditional back-propagation method, is built on top of DBN for supervised prediction. To our best knowledge, this is the first paper that applies the echo state network methodology to deep learning. The resulted model, combining the merits of DBN and ESN, provides a more robust alternative to conventional deep neural networks for the superior prediction capacity. Extensive experimental results show that our DBEN can achieve a significant enhancement in the prediction performance, learning speed, and short-term memory capacity.

© 2017 Elsevier B.V. All rights reserved.

1. Introduction

Currently, cognitive neuroscience discoveries [1–5] have provided a profound understanding of the principles on how to govern information representation in the mammal brain, developing new ideas of designing a system to effectively represent information. One of the crucial discoveries is that the mammal brain is organized in a deep architecture, in which the input percept can be represented by means of multiple levels of abstraction, each level corresponding to a different area of the cortex. Based on the learnt abstract features, human is capable of performing a wide variety of real-world tasks such as object detection, image recognition, prediction, and visualization. Hence, the machine learning community expresses the tremendous interest in this hierarchical architecture of the mammal brain, and attempts to mimic it in order to enhance the performance of a learning algorithm.

Currently, deep learning [6,7] has become one of the most sophisticated machine learning techniques to learn perfect information representation that provides similar functionality to that of the mammal brain. It exerts a tremendous fascination on researchers for establishing hierarchical representations from data. A general deep architecture is built in layers, each of which consists of feature detector units. Lower layers extract simple features and inject into higher layers, which can successively perceive more abstract features. In particular, deep belief network (DBN) [8], proposed by Hinton et al., is a powerful hierarchical generative model for feature extraction. Compared with the training methods of traditional deep models, such as multilayer perceptron, DBN can effectively obviate over-fitting to the training set by means of a distinctive unsupervised pre-training. Compared with the traditional shallow models, such as support vector machine (SVM), DBN can express highly variant functions, discover the potential laws existing in multiple features, and have a better generalization capacity, since “functions that can be compactly represented by a depth \( k \) architecture might require an exponential number of computational elements to be represented by a depth \( k − 1 \) architecture” [7]. Furthermore, Deng et al., defined a hybrid deep neural network (DNN) that was built by adding a discriminative model component over DBN, namely DBN-DNN [9]. In a typical DBN-DNN, DBN serves as feature extraction, while the back-propagation (BP) algorithm in discriminative model component is used to fine-tune the whole network. This hybrid architecture has been successfully applied to many real-world classification problems from various domains, such as hand-written character recognition [10], acoustic modeling [11], health state diagnosis [12], hyperspectral data classifica-
tion [13], fingerprint liveness detection [14], natural language understanding [15], information retrieval [16].

Recently, the hybrid DBN-DNN also shows great potential for time series prediction. For example, Kuremoto et al. [17] applied the typical DBN-DNN to time series prediction. Huang et al. [18] proposed a deep architecture consisting of a traditional DBN and a multitask regression layer for traffic flow prediction. In order to accelerate the learning, Shen et al. [19] introduced a conjugate gradient method to DBN-DNN for exchange rate prediction. Although the proposals have exhibited favorable prediction performance for various considered tasks, the used BP algorithm in the logistic regression (LR) layer (i.e., discriminative model component), that performs the global weight fine-tuning, limits their further development in time series prediction. On the one hand, massive iterative computation results in a slow convergence rate (i.e., learning speed). On the other hand, the BP algorithm based on gradient descent can be easily trapped in to a local optimum [20,21], leading to an unsatisfactory prediction accuracy. Thus further studies are required to explore more effective regression method instead of the much-maligned BP algorithm for superior prediction performance.

Echo state network (ESN), proposed by Jaeger et al. [22,23], is a typical paradigm of recurrent neural networks (RNNs). It is viewed as a powerful tool to model temporal correlations between the input and output sequences. The learning can be realized through offline linear regression or online methods [22–25], such as the recursive least square, providing optimal weights for the given ESN. ESNs offer some attractive advantages over classic RNNs using the BP algorithm, such as the faster training speed and stronger non-linear approximation capacity [25–29]. Inspired by it, we investigate the possibility of combining the training method of ESN and feature extraction of DBN so that the resulting model is able to achieve superior prediction performance.

In this paper, we propose a novel deep prediction framework, termed the deep belief echo-state network (DBEN). In the new structure, a DBN is connected to an ESN-based regression layer. For the learning of DBEN, the contrastive divergence method is used to train multiple RBMs in DBN, followed by the local weight adjustment in the LR layer via a supervised echo-state mechanism. The contribution of this paper could be summarized in three aspects.

1. To the best of our knowledge, this is the first work with implementation details of combining the deep learning and ESN methodology. The merits of both DBN, in terms of the high-efficient feature extraction from data, and ESN, in terms of the exceptional performance in modeling dynamical data, make DBEN a promising method for time series prediction.

2. The efficacy of the proposed approach is evaluated considering a number of widely used time series benchmarks, and compared with the baseline models, such as the DBN-based method, classical ESN and SVM, to demonstrate its superior performance.

3. We define the short-term memory (STM) of DBEN as the capacity that it can reconstruct the inputs of visible layer from the outputs, and further show the relationship between the STM and prediction performance.

The remaining of the paper is organized as follows. Section 2 provides some basic background on RBM, DBN-DNN and ESN. Section 3 elaborates the architecture and learning algorithm of DBEN. Experiments and evaluation results on the benchmark datasets are given in Section 4 with respect to prediction accuracy and STM. We discuss the proposed DBEN in Section 5. Finally, this paper is concluded in Section 6.

2. Theoretical background

In this section, we briefly summarize the theoretical background of the considered models, namely RBM, hybrid DBN-DNN and ESN, which are the basis of the following DBEN understanding.

2.1. RBM

RBM [7,16] is an energy-based stochastic neural network composed by two parts, i.e., visible layer and hidden layer, in which the training process is done in an unsupervised mode. Similar to the classical Boltzmann machine, the visible layer and hidden layer are fully connected via symmetric undirected weights, except that there exist no intra-layer connections within either the visible or hidden layer. The architecture of a typical RBM model is depicted in Fig. 1, where $v$ denotes the visible layer, $h$ denotes the hidden layer, and $w_{ij}$ denotes the connection weight between the visible unit $i$ and hidden unit $j$.

The surprising advantage of RBM is embodied in the idea of reconstruction oriented learning. Just the information in hidden units, learnt as features, can be used to reconstruct the input. Once the original input is recovered perfectly during reconstruction, it implies that the hidden units reserve input information as much as possible, and the updated weights and biases are capable of effectively measuring the input data.

2.2. DBN-DNN

The hybrid DBN-DNN [9] consists of both generative and discriminative model components corresponding to a traditional DBN and a LR layer, respectively, as shown in Fig. 2. Here, DBN is a probabilistic generative model with a stack of restricted Boltzmann machines (RBMs) [78,13–15], which can effectively model the structure in the input data (feature representation) [30]. In order to solve classification and regression problems, a LR layer is further added above DBN for supervised learning. Accordingly, the training of DBN-DNN includes two phrases: pre-training and fine-tuning.

In the first phrase, DBN is pre-trained in a greedy layer-wise unsupervised fashion, starting from the lowest RBM and using its outputs (after the current RBM training is completed) as the inputs for training the following RBM. Once the training of all RBMs is finished, the outputs of the final one is the learnt features in the pre-training procedure. It is widely agreed that the fast and effective pre-training is extremely beneficial to deep learning [9,30,31]. Then, in the second phrase, the BP algorithm is employed to fine-tune the whole pre-trained network to integrate the layers of neural networks by utilizing the learnt features. It can find a minimum in a peripheral region of parameters initialized by DBN [13]. Finally, the trained DBN-DNN can be used for classification and regression.

2.3. Echo state network

Echo state networks (ESN) establish an efficient and powerful approach to recurrent neural network (RNN) training. Unlike the traditional RNNs, such as Elman networks, that are organized in layers and contain feedback connections. The core part of ESN is a single reservoir consisting of a mass of neurons that are randomly interconnected and/or self-connected. The reservoir itself remains unchanged, once it is selected. The efficient learning can be achieved by determining the weights of the connections between the reservoir and the output layer. ESNs overcome the slow convergence combined with high computational requirements and suboptimal estimates of the model parameters, shown by RNN
training algorithms based on direct optimization of the network weights.

The supervised ESN training comprises two crucial steps. In the first step, the reservoir is driven by the input signals, and then the input signals are mapped onto a high-dimensional state space by a nonlinear transformation. Once the reservoir is activated, the weights of connections to an output layer are determined by a simple linear regression method. Further details with related to ESNs can be found in the literatures [22–24].

3. Deep belief echo-state network

In the section, we present a new deep architecture for time-series prediction, DBEN, and give the details of the whole architecture and learning algorithm.

3.1. DBEN architecture

In this section, we integrate the DBN methodology and echo state mechanism together to construct a novel deep prediction architecture, i.e., DBEN. The whole network architecture of DBEN consists of two key parts: a DBN with multiple RBMs and a novel regression layer, as shown in Fig. 3.

DBN serves as the nonlinear feature transformation. Features learned in the top layer of DBN are the most representative features for modeling the time-series data. They can be described by $H_p = h_{p1}, h_{p2}, \ldots, h_{pn}$, where $p$ denotes the top layer, and $n$ is the number of features in the top layer. It can be seen clearly that the most representative feature $H_p$ is the current state of last RBM. Then these obtained features are used as the input vector of the following DBEN component for regression. Since we employ an ESN-based regression layer in the top, DBEN can be viewed as a complete paradigm of a typical neural network. And from the groundbreaking architecture, DBN is a feature learning model, and the novel regression layer is a regression model. Likewise, the DBEN learning is also divided into two processes, namely the independent DBN learning and the local weight adjustment.

3.2. Independent DBN learning

Similar to the aforementioned pre-training, the independent DBN learning is also performed in a greedy layer-wise fashion. However, DBN itself is fixed, once the DBN training is achieved. In other words, the DBN learning in DBEN is not affected by the training in the next phase. Then, we give a systematic description of the independent DBN learning as follows.

Consider that each RBM has only binary visible and hidden units, which exhibit a special type of Markov random field with visible units $v = \{0,1\}^2$ and hidden units $h = \{0,1\}^2$. The energy of the joint configuration of the units, determined by the weights and biases of RBM, is given by

$$E(v, h; \theta) = -\sum_{i=1}^{D} a_i v_i - \sum_{j=1}^{F} b_j h_j - \sum_{i=1}^{D} \sum_{j=1}^{F} w_{ij} v_i h_j$$

$$= -a^T v - b^T h - v^T W h$$  \hspace{0.5cm} (1)

where model parameters $\theta = \{a_i, b_j, w_{ij}\}$. $w_{ij}$ denotes the weight between visible unit $i$ and hidden unit $j$. $a_i$ and $b_j$ are bias parameters of visible and hidden unit, respectively. The probability of a visible-hidden vector pair $(v, h)$ related to the energy function is defined as follows

$$p(v, h; \theta) = \frac{1}{Z(\theta)} \exp(-E(v, h; \theta))$$  \hspace{0.5cm} (2)

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

$$Z(\theta) = \sum_{v} \sum_{h} \exp(-E(v, h; \theta))$$  \hspace{0.5cm} (3)

RBM gives a probability to every input vector via the energy function. In order to lower the energy as shown in Eq. (1), the probability of the training vector can be improved by adjusting $\theta$.

The conditional probabilities corresponding to the hidden unit $j$ and visible unit $i$ are given by logistic function

$$p(h_j = 1|v) = \sigma(b_j + \sum_{i} v_i w_{ij})$$  \hspace{0.5cm} (4)

$$p(v_i = 1|h) = \sigma(a_i + \sum_{j} h_j w_{ij})$$  \hspace{0.5cm} (5)

where $\sigma(\cdot)$ stands for the logistic sigmoid function, that is, $\sigma(x) = 1/(1+\exp(-x))$. 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. (5). Then, the states of the hidden units are updated accordingly, so that they represent the reconstruction features.

The learning of $w$ is achieved by a contrastive divergence (CD) method [8]. The update rule in a weight is given by

$$\Delta w_{ij} = \eta (\langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{rec}})$$  \hspace{0.5cm} (6)
where $\eta$ is the learning rate. Through the learning process, we can obtain proper value of $w$. Repeating the process until the weights of all RBMs are updated, we can obtain a trained DBN.

3.3. Local weight adjustment

Obviously, the ESN-based regression layer is corresponding to a typical ESN paradigm. Different from the traditional fine-tuning based on the BP algorithm, a local weight adjustment (LWA) is performed in the ESN-based regression layer by using the learnt features (i.e., the output states $u(t)$ of last trained RBM, marked in green in Fig. 3). In fact, LWA is a supervised ESN training, conducted by updating the reservoir state and network output as follows

$$
    x(t + 1) = f(W^{in} u(t + 1) + W x(t) + W^{back} y(t))
$$

(7)

$$
    y(t + 1) = W^{out} x(t + 1)
$$

(8)

where $u(t)$ is the input signal fed to the reservoir at time $t$, i.e., learnt features, $x(t)$ is the reservoir state at time $t$, $y(t)$ is the network readout at time $t$, $f(\cdot)$ denotes the activation function of reservoir neurons (tanh in our case), $W^{in}$ denotes the input weight matrix, $W$ denotes the internal weight matrix of reservoir, $W^{back}$ denotes the feedback weight matrix, $W^{out}$ denotes the linear output weight matrix. Elements of $W^{in}$, $W$ and $W^{back}$ are generated randomly during the initialization with random values drawn from a uniform distribution, and fixed until the end of DBN execution. Only the output weight matrix $W^{out}$ needs to be trained. To account for echo state property, the internal weight matrix $W$ is typically scaled as

$$
    W \leftarrow \frac{\alpha}{|\lambda_{\text{max}}|} W
$$

(9)

where $|\lambda_{\text{max}}|$ is the spectral radius of $W$, and $\alpha \in (0, 1)$ is a scaling parameter. Additionally, due to the influence of initial reservoir states, a certain number of initial steps need to be abandoned during the training, called washout phase.

Essentially, LWA is to obtain the optimal $W^{out}$. During the process, the obtained reservoir states are collected in a state matrix $X$.

$$
    X = \begin{bmatrix}
    x^T(1) \\
    x^T(2) \\
    \vdots \\
    x^T(n)
    \end{bmatrix}
$$

(10)

and the corresponding target outputs are collected in a target output matrix $Y$

$$
    Y = \begin{bmatrix}
    y(1) \\
    y(2) \\
    \vdots \\
    y(n)
    \end{bmatrix}
$$

(11)

where $n$ denotes the number of training samples. Then, a linear regression problem should be solved for $W^{out}$, expressed as

$$
    XW^{out} = Y
$$

(12)

Generally, the least squares solution is used to solve the problem, given by

$$
    W^{out} = \arg \min_w \|Xw - Y\|
$$

(13)

where $\|\cdot\|$ denotes the Euclidean norm. The output weight matrix $W^{out}$ can be solved in a single step using the Moore-Penrose pseudo inverse

$$
    W^{out} = \bar{X}Y = (X^TX)^{-1}X^TY
$$

(14)

where $\bar{X}$ denotes the generalized inverse of $X$. It means the completion of LWA in regression layer. Algorithm 1 depicts the pseudo-code of the LWA algorithm. Afterwards, our DBN is used to perform various nonlinear approximation tasks.

4. Experiments

In the following section, we provide a comprehensive experimental evaluation for the DBEN model, considering four time series benchmark: the nonlinear autoregressive moving average (NARMA) system, the Mackey-Glass (MG) system, the multiple superimposed oscillator (MSO) problem and the sunspots series. In our experiments, each dataset is divided into three parts for training, validating, and testing, respectively. To illustrate the advantages of our approach, we also evaluate DBN-based method
Algorithm 1: LWA algorithm.

Input: \( u(t) \), \( x(t) \) and \( y(t) \) are the input vector, reservoir state and output vector at time \( t \), respectively;
\( \lambda \) is the spectral radius;
\( W^m \), \( W \) and \( W^\text{back} \) are input, reservoir and feedback weight matrices, respectively;
\( T \) is the collection time on the reservoir state, \( T_0 \) is the washing time;
\( h^l \) is the output states of the last RBM.
Output: \( W^\text{out} \) is the output weight matrix.

1. **Network Initialization**
   1. \( W^m, W, W^\text{back} \) are generated randomly;
   2. \( \lambda \in (0, 1) \);
   3. \( x(0) = 0 \);
2. **Sampling Network Training Dynamics**
   1. \( u(t) \leftarrow h^l \);
   2. for \( t = 0 \) to \( T \) do
   3. \( \) update the reservoir states using Eq. (7);
   4. \( \) collect the reservoir state in the state matrix \( X \);
   5. \( \) collect the network output in the output matrix \( Y \);
3. **Output weights computation**
   1. Solve \( W^\text{out} \) using Eq. (14).

(DBN) [17], classical echo state network (CESN) [23], and SVMs with the kernels of multilayer perception and radial basis function (SVM, MLP and SVM, RBF). Table 1 depicts the parameter configurations of the evaluated models. In particular, DBEN and DBN are evaluated considering 5 iterations of the CD algorithm, i.e., CD-5. In the experiments, the normalized root-mean-square error (NRMSE) is used to measure the prediction accuracy of the evaluated models, expressed as follows

\[
\text{NRMSE} = \sqrt{\frac{\sum_{t=1}^{l_{\text{test}}} (y_{\text{test}}(t) - d(t))^2}{l_{\text{test}} \cdot \sigma^2}} \tag{15}
\]

where \( l_{\text{test}} \) is the length of test samples, \( y_{\text{test}}(t) \) and \( d(t) \) are the test output and desired output at time step \( t \), respectively, and \( \sigma^2 \) is the variance of desired output \( d(t) \). All the simulations are carried out in an identical software and hardware environment, and for each task, the average results over 50 trials are obtained for comparison.

4.1. NARMA system

The NARMA system [32] is a discrete time system, in which the current output depends on both the input and the previous output. In general, it is really difficult to model this system, due to the nonlinearity and possibly long memory. In the prediction task, the NARMA time series is generated from the following 10th-order NARMA system

\[
y(n) = 0.3y(n-1) + 0.05y(n-1) \sum_{i=1}^{10} y(n-i) + 0.1
\]

where \( y(n) \) is the system output at time \( n \), and \( x(n) \) is the system input at time \( n \), randomly drawn from a uniform distribution over the interval \([0, 1]\). In our system identification task, 15,000 points are generated by (16) and divided into three parts: the first 6000 values are used for training, the next 3000 are used as a hold-out validation, and the remaining 6000 are used for testing. The first 100 values of training and testing sequences are discarded to wash out the initial transient. DBEN and DBN contains two hidden layers, and the sizes are set to 100 and 200, respectively.

Fig. 4 shows the prediction results over a selected region from time step 2600–2700 obtained by using the considered models for 10th order NARMA task. As we can observe from this figure, DBEN is able to offer a most accurate prediction of the target signal, while DBN, SVM, MLP and SVM, RBF are not completely powerless to reconstruct the target signal. The prediction NRMSEs for DBEN, DBN, CESN, SVM, MLP and SVM, RBF are 0.1572, 1.0024, 0.4103, 1.0035 and 1.0005 respectively.

To investigate the effects of deep learning parameters on the prediction accuracy, we give the prediction NRMSEs of DBEN and DBN on the CD-\( k \), leaning rate \( \alpha \) and batch-size \( \beta \) in Fig. 5 and Tables 2 and 3, respectively. As we observe, DBEN is superior to DBN over the choice of these parameters for the 10th order NARMA task. It is noteworthy that the prediction performance of our model is not satisfactory at the beginning of iteration, but after several iteration steps, the performance starts to get better. Moreover, the relatively small \( \alpha \) and \( \beta \) are conducive to good prediction performance. In Fig. 6, we plot the prediction NRMSEs of the DBEN and CESN models as functions of reservoir size and spectral radius. From the figure, we observe that DBEN performs better than CESN. Especially, DBEN is more sensitive to changes in the parameter \( \lambda \). The region where DBEN shows the best prediction accuracy lies between \( \lambda = 0.6 \) and \( \lambda = 0.9 \). Moreover, when the reservoir size is relatively small, DBEN is still able to offer a good prediction accuracy. Whereas, CESN is entirely dependent on changes in the parameters \( N \) and \( \lambda \), and tends to perform well as both parameters increase.

4.2. Mackey-Glass System

The MG system [26,29] is a classical benchmark with a chaotic attractor for time series modeling. It has been widely applied to the performance testing of models for prediction in many literatures. The MG time series is derived from a time-delay differential system with the following form

\[
\frac{dy(t)}{dt} = ay(t-\tau) + by(t) \tag{17}
\]

where the related parameters are set as \( n = 10 \), \( a = 0.2 \), and \( b = -0.1 \). When the delay time \( \tau < 16.8 \), the MG system can exhibit chaotic behavior. In our experiment, 10,000 points are generated from Eq. (17) for delay time \( \tau = 17 \), similar to [29] and divided into three parts: the first 4000 values are used for training, the next 2000 are used as a hold-out validation, and the remaining 4000 are used for testing. The first 100 values of training and testing sequences is discarded to wash out the initial transient. DBEN and DBN contains two hidden layers, and the sizes are set to 10 and 50, respectively.

In Fig. 7, we show comparative trajectories of the original MG and ones produced by the evaluated models. As seen from this figure, the trained DBEN comes closer to the original MG attractor than the other trained models. The corresponding prediction NRMSEs are 0.00048, 0.4435, 0.0788, 0.1448 and 0.1449, respectively.

For the MG dynamic system, if the time delay becomes larger, the system will exhibit the more severe nonlinearity. In particular, the chaos emerges in the MG system when \( \tau > 16.8 \). Therefore, it is extremely difficult to approach such nonlinear dynamic system using the existing models with the increase of \( \tau \). Undoubtedly, it is a serious challenge. Table 4 depicts the performance of the evaluated models in terms of the time delay \( \tau \). Our results indicate that DBEN works better than the considered alternatives. Even though \( \tau \) rises to 27, which means that system nonlinearity becomes more severe and makes it rather hard to deal with this problem, our
To investigate the effect of the reservoir parameters on prediction performance, we plot the prediction NRMSEs of the three network structures as a function of the reservoir size and spectral radius in Fig. 8. As we observe, DBEN and CESN show similar relations among the prediction NRMSE, reservoir size and spectral radius, that is, the larger the N and λ is, the better the prediction performance of both models are. Moreover, it is noteworthy that DBEN significantly outperforms CESN over the choices of N and λ, and is capable of achieving the surprising prediction accuracy using a small quantity of reservoir neurons. For example, when N = 20 and λ = 0.1, the prediction NRMSE of DBEN is just 0.00229, while CESN can achieve such accuracy only if the reservoir size is increased up to 190 and beyond.

4.3. Multiple superimposed oscillator problem

In this subsection, we study the performance of DBEN in solving the MSO problem [33]. The considered MSO time series data

---

**Table 1**

Configuration of the evaluated models in our experiments.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>NARMA system</th>
<th>Mackey-glass system</th>
<th>MSO</th>
<th>Sunspots</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning rate α</td>
<td>0.01</td>
<td>0.05</td>
<td>0.001</td>
<td>0.05</td>
</tr>
<tr>
<td>Batch-size β</td>
<td>100</td>
<td>50</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>Reservoir size N</td>
<td>200</td>
<td>50</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>Spectral radius λ</td>
<td>0.6</td>
<td>0.7</td>
<td>0.6</td>
<td>0.8</td>
</tr>
<tr>
<td>Output feedback</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Output self-feedback</td>
<td>x</td>
<td>x</td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>Type of output neuron</td>
<td>Linear</td>
<td>Linear</td>
<td>Linear</td>
<td>Linear</td>
</tr>
<tr>
<td>Regularization parameter γ</td>
<td>1.5</td>
<td>3.5</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Kernel parameter σ</td>
<td>1</td>
<td>1.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

---

**Table 2**

Performance of DBEN and DBN with different learning rate α for the 10th order NARMA task.

<table>
<thead>
<tr>
<th>Model</th>
<th>0.02</th>
<th>0.04</th>
<th>0.06</th>
<th>0.08</th>
<th>0.1</th>
<th>0.12</th>
<th>0.14</th>
<th>0.16</th>
<th>0.18</th>
<th>0.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBEN</td>
<td>0.1302</td>
<td>0.1036</td>
<td>0.1678</td>
<td>0.2424</td>
<td>0.3722</td>
<td>0.3781</td>
<td>0.3774</td>
<td>0.3787</td>
<td>0.3761</td>
<td>0.3762</td>
</tr>
<tr>
<td>DBN</td>
<td>1.0041</td>
<td>0.9999</td>
<td>1.0028</td>
<td>1.0052</td>
<td>0.9991</td>
<td>1.0077</td>
<td>0.9993</td>
<td>1.0001</td>
<td>0.9970</td>
<td>1.0049</td>
</tr>
</tbody>
</table>

---

**Table 3**

Performance of DBEN and DBN with different batch-size β for the 10th order NARMA task.

<table>
<thead>
<tr>
<th>Model</th>
<th>50</th>
<th>100</th>
<th>150</th>
<th>200</th>
<th>250</th>
<th>300</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBEN</td>
<td>0.0894</td>
<td>0.1572</td>
<td>0.1626</td>
<td>0.3340</td>
<td>0.3647</td>
<td>0.3634</td>
</tr>
<tr>
<td>DBN</td>
<td>1.0169</td>
<td>1.0024</td>
<td>0.9995</td>
<td>1.0031</td>
<td>1.0039</td>
<td>1.0004</td>
</tr>
</tbody>
</table>

---

**Table 4**

Performance of DBEN, DBN, CESN, SVM_MLP and SVM_RBF versus the time delay τ in the MG system.

<table>
<thead>
<tr>
<th>Model</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
<th>23</th>
<th>24</th>
<th>25</th>
<th>26</th>
<th>27</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBEN (×10⁻⁴)</td>
<td>5.04</td>
<td>4.98</td>
<td>5.08</td>
<td>5.36</td>
<td>4.91</td>
<td>5.05</td>
<td>5.47</td>
<td>5.45</td>
<td>5.50</td>
<td>5.53</td>
</tr>
<tr>
<td>DBN</td>
<td>0.5338</td>
<td>0.4310</td>
<td>0.5155</td>
<td>0.4914</td>
<td>0.5065</td>
<td>0.5233</td>
<td>0.4871</td>
<td>0.4614</td>
<td>0.5137</td>
<td>0.4552</td>
</tr>
<tr>
<td>CESN</td>
<td>0.0731</td>
<td>0.0897</td>
<td>0.1172</td>
<td>0.1311</td>
<td>0.1317</td>
<td>0.1375</td>
<td>0.1563</td>
<td>0.1752</td>
<td>0.1864</td>
<td>0.2186</td>
</tr>
<tr>
<td>SVM_MLP</td>
<td>0.1403</td>
<td>0.1357</td>
<td>0.1303</td>
<td>0.1278</td>
<td>0.1256</td>
<td>0.1226</td>
<td>0.1216</td>
<td>0.1208</td>
<td>0.1167</td>
<td>0.1155</td>
</tr>
<tr>
<td>SVM_RBF</td>
<td>0.1407</td>
<td>0.1359</td>
<td>0.1306</td>
<td>0.1281</td>
<td>0.1259</td>
<td>0.1229</td>
<td>0.1219</td>
<td>0.1213</td>
<td>0.1169</td>
<td>0.1160</td>
</tr>
</tbody>
</table>
are generated by summing up several simple sine wave functions. Formally it is expressed as the following equation

$$y(n) = \sum_{i=1}^{s} \sin(\alpha_i n)$$  \hspace{1cm} (18)

where $s$ denotes the number of sine waves, $\alpha_i$ denotes the frequencies of the summed sine waves, and $n$ specifies an integer index of the time step. Subsequently, we use MSOx to describe especial MSO dynamics, where $x$ defines the number of summed sine waves. The MSO problem with different numbers of sine waves has been considered in the literatures [29,33], where the frequencies of the sine waves are taken from the same set: $\alpha_1 = 0.2$, $\alpha_2 = 0.311$, $\alpha_3 = 0.42$, $\alpha_4 = 0.51$, $\alpha_5 = 0.63$, $\alpha_6 = 0.74$, $\alpha_7 = 0.85$ and $\alpha_8 = 0.97$. In the experiment, the used MSO sequence has a length of 5000 items where the first 2000 items were used for training, the next 1000 were used as a hold-out validation set and the remaining 2000 for testing. DBEN and DBN contains two hidden layers, and the sizes are set to 5 and 10, respectively.

Firstly, MSO2, composed of two sines: $y(n) = \sin(0.2n) + \sin(0.311n)$ $n = 1, 2, \ldots$ is used to train the evaluated models. Fig. 9 shows comparative curves of the network output and predicted signal versus time steps $n$ as well as the corresponding output error for each model under consideration. We see that DBEN again demonstrates its ability to model the nonlinear system with high performance, using an ESN-embedding strategy. The prediction NRMsEs for DBEN, DBN, CESN, SVM_MLP and SVM_RBF are 0.00038, 0.7278, 0.6706, 0.0323 and 0.0129, respectively. The improved performance of DBEN is further illustrated in Table 5. As expected, DBEN significantly performs better than the other evaluated models, even in the face of the MSO task with numerous sine waves. In general, the simulation results from Table 5 indicate that DBEN can solve the MSO problem well.

### 4.4. Sunspots series prediction

The sunspot series can intuitively reflect the activity level of solar that brings enormous impact to the Earth. However, it is an extremely severe challenge to predict the sunspot series, due to the data complexity and lack of a mathematical model. Hence, we use the sunspot series to validate the performance of DBEN on solving real-world problems. Two kinds of dataset on sunspot number are used in this experiment, i.e., smoothed and jerky sunspot number data [29], which contain 3174 sunspots numbers from January 1749 to

![Fig. 5. Performance of DBEN and DBN on CD iterations for the 10th order NARMA task.](image)

![Fig. 6. Dependency of network performance on chosen reservoir size $N$ and spectral radius $\lambda$ for the 10th order NARMA task.](image)

<table>
<thead>
<tr>
<th>Table 5</th>
<th>Performance of DBEN, DBN, CESN, SVM_MLP and SVM_RBF for the different MSO tasks.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>MSO3</td>
</tr>
<tr>
<td>DBEN</td>
<td>0.0023</td>
</tr>
<tr>
<td>DBN</td>
<td>0.7751</td>
</tr>
<tr>
<td>CESN</td>
<td>0.5372</td>
</tr>
<tr>
<td>SVM_MLP</td>
<td>0.0558</td>
</tr>
<tr>
<td>SVM_RBF</td>
<td>0.0170</td>
</tr>
</tbody>
</table>
to November 2014, respectively, with $L_{\text{trn}} = 3000$, $L_{\text{val}} = 1500$ and $L_{\text{tst}} = 3000$. DBEN and DBN contains two hidden layers with the size of 3 neurons.

Figs. 10 and 11 show prediction outputs and errors obtained by DBEN over the whole test region for smoothed and jerky sunspot time series. It is shown that our model can predict smoothed sunspot series well. The margin of error ranges from $-2.5$ and $2.5$ in most points. However, due to the influence of strong noise contained within jerky sunspot series, our model shows the significant degradation of prediction performance. The margin of error ranges from $-50$ and $50$ in most points. The performance of each structure under consideration is further listed in Table 6. As we observe, DBEN can effortlessly beat the considered alternatives in the real-world sunspots series tasks.

Table 6

<table>
<thead>
<tr>
<th>Model</th>
<th>DBEN</th>
<th>DBN</th>
<th>CESN</th>
<th>SVM_MLP</th>
<th>SVM_RBF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunspots-S</td>
<td>0.0231</td>
<td>1.6412</td>
<td>0.5303</td>
<td>0.0682</td>
<td>0.2881</td>
</tr>
<tr>
<td>Sunspots-J</td>
<td>0.3439</td>
<td>1.5277</td>
<td>0.5980</td>
<td>0.3690</td>
<td>0.3755</td>
</tr>
</tbody>
</table>
4.5. Short-term MC

In order to investigate the temporal processing ability of proposed deep learning architecture, we further focus on how much past input information of DBEN can be encoded in the instantaneous spatial state of the system. The property is defined by memory capacity (MC), just a measure of the ability to reconstruct the input signal from a past time \( t \), as illustrated in [34] by Jaeger. For the sake of argument, we assume that the network is actuated by an univariate stationary input signal \( u(t) \). For a given delay \( k \), consider the task of outputting \( u(t - k) \) after seeing the i.d.d. input stream \( \ldots u(t - 2)u(t - 1)u(t) \) up to time \( t \) for DBEN. The well-fitting characteristic is measured in terms of the squared correlation coefficient between the desired output (i.e., input signal delayed by \( k \) time steps) and the observed network output \( y(t) \)

\[
MC_k = \frac{Cov(u(t-k), y(t))}{Var(u(t))Var(y(t))}
\]  

(19)

where \( Cov \) denotes the covariance, and \( Var \) denotes variance, \( u(t) \) and \( y(t) \) is the network input and output of DBEN at time step \( t \), respectively. The short-term memory (STM) capacity is then formulated as:

\[
MC = \sum_{k=1}^{\infty} MC_k
\]

(20)

We empirically evaluate the short-term MC of DBEN and CESN. The networks are trained to memorize the inputs delayed by \( k = 1, 2, \ldots, 40 \). Here, consider that DBEN uses the network structure of 1-10-20-40, whose parameters are set as follows: \( \alpha = 0.004 \) and \( \beta = 100 \). Apparently the reservoirs contain 20 neurons. The MC of the DBEN model is evaluated using the 2000 samples of the 10th-order NARMA dataset in Section 4.1, and initial transient is washed out by employing a reservoir warm-up time of 100 steps.

In Fig. 12, we plot the forgetting curves of evaluated models for different spectral radius \( \lambda \), where the detCoeff is the squared correlation coefficient (i.e., \( MC_1 \) in Eq. (19)). One obvious fact is clearly that DBEN shows more powerful \( k \)-delay memory capacities than...
the CESN over the choice of \( \lambda \). The next interesting fact is that the trend of \( \text{MC}_k \) depends heavily on the spectral radius. As \( \lambda \) increases, the evaluated models possess a better memory. Specially, when \( \lambda = 0.9 \), DBEN exhibits a close-to-100% recall for delays up to 13, followed by a craggy slope that still is visibly above zero until the delay rises to 40. Furthermore, the STM capacities of DBEN and CESN are shown in Table 7, where we see that DBEN has more excellent STM capacity, particularly for bigger \( \lambda \). We also plot the reconstruction NRMSEs of the evaluated models for all delays in Fig. 13. As we observe, DBEN significantly outperforms CESN for each reconstruction ranged from 1 to 40 delays, considering different \( \lambda \). In order to measure the nonlinear approximation capability of the evaluated models, we average the NRMSEs across the finite regions of \( \lambda \), as shown in Table 8, where the experimental results again demonstrates its ability of DBEN to reconstruct the network inputs with high performance. For example, for \( \lambda = 0.9 \), our model increases the reconstruction performance to around 69%, which is 20% higher than that obtained by CESN. Besides, combined with Table 7, we give the interesting relationship that the more powerful the STM capacity is, the better nonlinear approximation capacity DBEN has.

### 5. Discussion

In the study, our objective is to investigate the representational power of the newly proposed deep belief echo-state network in
time series prediction. The incorporation of the LWA method based on the typical ESN paradigm offers us a novel problem-solving capability, i.e., low computational complexity and marvelous prediction accuracy. The experimental results over the datasets allow us to draw some important conclusions.

(1) As a promising deep architecture, our DBEN greatly outperforms the deep model (DBN) and the shallow models (CESN, SVM_MLP and SVM_RBF) in prediction performance. Here, we would like to make two substantial comments related to the excellent prediction performance of DBEN. The first one concerns the feature extraction capacity. As we all know, DBN is actually regarded as hierarchical feature detectors that can capture complex statistical patterns in data, thus offering assistance to the following LWA. To give a better insight into feature extraction capacity, we evaluate the effect of network depth on the prediction performance in Fig. 14, where the corresponding parameter configuration is the same as Table 1 and the size of each hidden layer is set to 10. As we observe, DBEN with two hidden layers achieves better prediction performance in the benchmark tasks (NARMA, MG and MSO), but as the network depth increases, the prediction performance deteriorates gradually (see Fig. 14(a)–(c)). As for the real-world application (sunspots), DBEN obtains the superior prediction performance in the case that the network depth is up to 6 (see Fig. 14(d) and (e)). These results indicate that the feature extraction capacity of DBEN is conducive to improving the prediction performance, since the independent DBN learning provides the excellent starting points for the following LWA [30]. Our second comment regards LWA based on the echo-state mechanism. Theoretically, we can demonstrate the uniqueness of the generalized inverse $\hat{X}$ since “the pseudo-inverse of a matrix is unique” [35,36]. Furthermore, we conclude that the obtained $W_{out}$ is also unique for LWA, namely, a global optimum (see Eq. (14)). The two main characteristics enable our DBEN model to achieve excellent prediction performance, as shown in Figs. 4–11 and Tables 2–6.

(2) DBEN exhibits the surprising learning speed. Table 9 shows the training time of the evaluated models for the considered prediction tasks in Sections 4.1–4.4. We observe that DBEN has a rather lower overhead time than DBN and SVMs. This is mainly attributed to the following two aspects. On the one hand, unlike the global weight adjustment based on the BP algorithm, the used LWA independent of the preceding DBN learning just considers the determination of output weights $W_{out}$, which can effectively save the training time. On the other hand, the direct computation of $W_{out}$ (see Eq. (14)) rather than the massive iteration in the BP algorithm significantly reduces the training time as well. We also observe that DBEN is not as well as CESN in performing the training time, since the learning of stacked RBMs is much more time-consuming. Even so, our model is still a promising alternative to time series prediction, due to its outstanding nonlinear approximation capacity (see Figs. 4–11 and Tables 2–6). But in order to achieve such prediction performance, the shallow CESN model needs larger reservoir size and spectral radius. As a result, it leads to more overhead time. Taking the 10th order NARMA task for example, when the reservoir size is up to 850, CESN is still worse than DBEN (0.2302 and 0.1572, respectively.), and meanwhile its training time soars close to 10s far beyond DBEN (see Table 9). From the above discussion, we can see that DBEN offers a very good tradeoff between learning speed and prediction performance.

![Fig. 13. The reconstruction error curves of the evaluated methods for different $\lambda$ in the 10th order NARMA task. The spectral radius varies from 0.1 to 0.9.](image-url)

**Table 8**

Reconstruction performance of DBEN and CESN versus the spectral radius $\lambda$ in the 10th order NARMA task.

<table>
<thead>
<tr>
<th>Model</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBEN</td>
<td>0.8403</td>
<td>0.8059</td>
<td>0.7714</td>
<td>0.7540</td>
<td>0.7276</td>
<td>0.7091</td>
<td>0.6915</td>
<td>0.6917</td>
<td>0.6859</td>
</tr>
<tr>
<td>CESN</td>
<td>0.9742</td>
<td>0.9652</td>
<td>0.9561</td>
<td>0.9401</td>
<td>0.9257</td>
<td>0.9084</td>
<td>0.9016</td>
<td>0.9043</td>
<td>0.8960</td>
</tr>
</tbody>
</table>

**Table 9**

The training time (s) of the evaluated models.

<table>
<thead>
<tr>
<th>Model</th>
<th>NARMA</th>
<th>MG</th>
<th>MSO</th>
<th>Sunspots-S</th>
<th>Sunspots-J</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBEN</td>
<td>4.27</td>
<td>0.70</td>
<td>0.36</td>
<td>0.78</td>
<td>0.84</td>
</tr>
<tr>
<td>DBN</td>
<td>4.95</td>
<td>2.37</td>
<td>0.86</td>
<td>3.32</td>
<td>3.35</td>
</tr>
<tr>
<td>CESN</td>
<td>0.95</td>
<td>0.22</td>
<td>0.13</td>
<td>0.19</td>
<td>0.12</td>
</tr>
<tr>
<td>SVM_MLP</td>
<td>53.38</td>
<td>24.07</td>
<td>1.22</td>
<td>13.70</td>
<td>14.09</td>
</tr>
<tr>
<td>SVM_RBF</td>
<td>5.79</td>
<td>2.34</td>
<td>0.44</td>
<td>0.93</td>
<td>1.03</td>
</tr>
</tbody>
</table>
(3) **DBEN possesses the powerful STM capacity.** It is mainly attributed to the multi-layered RBMs in our DBEN that effectively capture data features, which further enhance the STM capacity on the basis of the reservoir. In fact, when DBEN is used to model highly nonlinear systems, the nonlinear approximation capability is directly dependent on its memory capacity, or more exactly, on its massive STM [37]. It has been illustrated in [34] that a larger spectral radius can cause a slower decay of the networks response to an impulse input, but a stronger networks memory capacity. In the case, ESN has better nonlinear approximation capacity. As expected, Figs. 12–13 and Tables 7 and 8 again verify the findings.

(4) **Despite these advantages, DBEN has an intractable problem on how to determine numerous parameters.** Except for the deep learning parameters, such as depth, neuron number, learning rate and batch-size, many reservoir parameters, such as reservoir size, spectral radius and connectivity, need to be determined in DBEN, because of the use of the ESN training method. These parameters have a strong impact on the prediction accuracy and computational complexity. Determining so many parameters is a challenging task [17,38]. Essentially, this is a multi-objective optimization problem, but beyond the scope of this paper. In this paper, all parameters are empirically selected for a given prediction task, and we just focus on whether DBEN has the superior performance or not.

**6. Conclusion and future work**

In the paper, we propose an extension of conventional DBN-based models under an echo-state perspective, namely, DBEN. It is constructed by integrating an ESN-based regression into the DBN architecture. Actually, the new regression layer is a network structure similar to ESN. Embedded in the DBEN’s structure, our new deep learning network just conducts a local weight adjustment by using the echo-state learning mechanism, independent of the preceding DBN learning. This novel way can effectively avoid the slow convergence and local optimum caused by the BP algorithm. Combined with the powerful feature extraction of DBN, our DBEN achieves the superior prediction performance. Experimental results on four widely used time-series benchmarks of different origins and characteristics demonstrate that DBEN can perform much better than DBN, CESN and SVMs in terms of prediction accuracy, STM, and training time. Future research will work on optimizing multiple parameters related to DBEN with optimization algorithms (e.g., particle swarm optimization and genetic algorithm).

**Acknowledgments**

This work was funded in part by National Natural Science Foundation of China (NSFC) under grant number 61503120 and 91646116, and Science and Technology Support Program of Jiangsu Province (No. BE2016776).

**References**

