Neurocomputing 275 (2018) 1333–1349

Contents lists available at ScienceDirect

Neurocomputing

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

Recurrent neural system with minimum complexity: A deep learning perspective

Xiaochuan Sun[a,b,*], Tao Li[b], Yingqi Li[a], Qun Li[b], Yue Huang[b], Jiayu Liu[a]

[a] School of Information Engineering, North China University of Science and Technology, Tangshan 063009, PR China
[b] Jiangsu BDSIP Key Lab, School of Computer science and Technology, Nanjing University of Posts and Telecommunications, Nanjing 210023, PR China

ARTICLE INFO

Article history:
Received 7 April 2017
Revised 4 July 2017
Accepted 17 September 2017
Available online 28 September 2017

Communicated by Deng Cheng

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

ABSTRACT

This paper proposes a novel echo state network (ESN) architecture in a deep learning framework for time series prediction. The architecture is a uniform and consistent system with functional parts of the pre-training input network that effectively captures information with different degrees of abstraction in the observed data, and the minimum complexity ESN that possesses the powerful nonlinear approximation capability and highly efficient training. To our best knowledge, this is the first systematic model attempting to introduce the deep learning methodology to the ESN modeling, which provides a more robust alternative to the conventional shallow ESNs. Extensive experiments on various widely used benchmarks of different origins and features show that our model achieves a great enhancement in the prediction accuracy and short-term memory capacity, without significant tradeoff in the model's computational efficiency.

1. Introduction

Recurrent neural network (RNN) offers an outstanding nonlinear approach for modeling dynamic systems, which is characterized by the recurrent connections between neurons. The special architecture is capable of directly processing temporal dependencies related to systems. Generally, RNNs can approximate arbitrary nonlinear dynamical system with arbitrary precision [1]. However, the early RNN architectures suffer from their limited memory capacity, due to the vanishing or exploding gradient problem [2], especially when the information involved in past inputs need to be recovered over a long time interval. In other words, the conventional RNNs poorly handle the long-term temporal dependencies. In contrast, the Long Short Term Memory (LSTM) architecture is capable of dealing with this problem by designing special memory cells that is actually a gated access mechanism to the neurons states [3]. Unfortunately, as a typical gradient-based network, LSTM is still unable to get rid of the drawbacks [2,4], such as slow convergence, excessive calculation and suboptimal solutions. These issues are mainly attributed to the unfolding and backpropagation through time procedure.

An extremely efficient network structure of RNN, called echo state network (ESN), was designed independently by Jaeger [5,6] for solving all the aforementioned issues. ESN is viewed as a powerful tool to model temporal correlations between the input and output sequences, whose kernel part is a single reservoir consisting of a great many neurons that are randomly interconnected and/or self-connected. The reservoir itself remains unchanged, once it is selected. During the ESN training, only the output weights need be computed through offline linear regression or online methods, such as the recursive least square [5–7], which considerably reduces computational complexity. Consequently, the ESN paradigm completely escapes these shortcomings of gradient-descent RNNs (e.g., LSTM) listed above. Until now, ESN has been successfully applied in various research areas, e.g., noise modeling [6], pattern recognition [8], robot control [9], reinforcement learning [10] and time series prediction [11–14].

As a result of these merits, ESN has captured widespread attentions of the computational intelligence community, and many ESN extensions have been explored. The existing ESN implementations mostly concentrate on the design of the network topologies, the selection of the neuron types, and the proposal of training algorithm. For example, Rodan and Tino [11] proposed a minimum complexity network structure for ESN, and through an exhaustive experimental and theoretical analysis, demonstrated that a reservoir could be simplified as much as possible, but not compromising the model's performance. Moreover, the most simplified ESN possessed the memory capacity being arbitrarily close to the

* Corresponding author at: School of Information Engineering, North China University of Science and Technology, Tangshan 063009, PR China.
E-mail address: sunxiaochuan@njupt.edu.cn (X. Sun).

https://doi.org/10.1016/j.neucom.2017.09.075
0925-2312/

proved optimal value. Qiao et al. [12] constructed a growing ESN with a multiple subreservoirs in an incremental way, leading to superior prediction performance and learning speed, and gave the proof of the convergence. Holzmann and Hauser [13] introduced general infinite impulse response filter neurons instead of original sigmoid ones as well as a delay&sum readout for ESNs, which significantly outperformed the standard ESN and other state-of-the-art models for nonlinear dynamical system modeling. Li et al. [14] proposed a robust ESN in a Bayesian framework that replaced the original linear regression with the Bayesian regression, so that the resulted model was capable of dealing with outliers in the training data set. Although these studies have given the remarkable advantages in the modeling performance, the extended ESN models are just shallow neural networks, characterized by a single nonlinear transformation of the input data into a feature space, followed by a linear mapping. A number of recent theoretical results have demonstrated that the shallow structures are insufficient at representing some functions [15–17], since they do not fully consider the diversity of the space distributions of the features in observed data, which greatly affects nonlinear approximation capability. Conversely, deep networks typically exhibit more powerful representation capacity than shallow networks with the same number of parameters for certain types of problems [18,19]. Hence, we would like to refine ESNs for superior nonlinear approximation capability from a deep learning perspective.

Generally, a deep architecture is built-in layers, each of which consists of feature detector units responsible for feature extraction [20,21]. Lower layers extract simple features and inject into higher layers, which successively perceive more abstract features. Deep belief network (DBN) [22–24] is one of the most important multiple-layer deep network architectures as well as a powerful probabilistic generative model. It can be trained to extract a deep hierarchical representation of input data by maximizing the likelihood of training data. Compared with the traditional shallow models, such as support vector machine, 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"* [15]. Especially, the DBN model, proposed by Hinton et al. [24], is a most promising alternative for deep networks. Structurally, it is viewed as stacked restricted Boltzmann machines (RBMs), each of which contains a visible layer representing observed data and a hidden layer learning to represent features that capture higher-order correlations in the data [25]. This promising architecture has been successfully applied to various domains, such as natural language understanding [18], hand-written character recognition [26], acoustic modeling [27], action recognition [28,29], super-resolution image processing [30,31], information retrieval [32]. Inspired by this, we investigate the possibility of introducing the DBN methodology to the ESN paradigm so that the resulting model is able to obtain more powerful nonlinear approximation capability.

In this paper, we propose a recurrent neural system with minimum complexity from a deep learning perspective. A unified and consistent architecture is built by integrating DBN and minimum complexity ESN, where a pre-training input network (PIN) is fully connected to the considered ESN structure. The special component is able to learn the multi-level features related to the modeled sequential data, providing a powerful support for the subsequent nonlinear approximation. Similar to DBN, PIN is also composed of stacked RBMs, whereas its learning in a greedy layer-wise mode based on contrastive divergence theory (CD) is completely irrelevant to the following ESN training. Extensive experiments demonstrate the superiority of the proposed model. In summary, the contributions of this paper include three aspects.

(1) To the best of our knowledge, this is the first attempt to introduce the deep learning methodology to the ESN modeling. In theory, the powerful capability of PIN on feature extraction can make the minimum complexity ESN a promising method for time series prediction.
(2) The short-term memory (STM) of our model is redefined as the ability of recovering the visible inputs from the whole network output, and the relationship between the STM capacity and the prediction performance is further given.
(3) The efficacy of the proposed model is evaluated considering a number of well-known benchmark datasets for prediction. Compared to the state-of-the-art models, our model obtains better prediction accuracy, while the computational burden is not significantly increased.

The remaining of the paper is organized as follows. Section 2 provides some basic background on RBM, and DBN. Section 3 elaborates the architecture and learning algorithm of the proposed model. Experiments and evaluation results on the benchmark datasets are given in Section 4. We empirically analyze the STM capacity of our deep ESN model in Section 5. We discuss our DSCR model in Section 6. Finally, this paper is concluded in Section 7.

2. Background

In this section, we give a brief introduction to the concepts related to DBNs, but particularly focus on the theoretical background of RBMs, which are the foundation of DBN understanding.

2.1. Restricted Boltzmann machines

A restricted Boltzmann machine [15,25,32] is a two-layer, undirected, bipartite graphical model composed by two parts, i.e., visible layer and hidden layer, which is trained in an unsupervised mode. Similar to the classical Boltzmann machine, the visible layer and hidden layer are fully connected via symmetric undirected weights, but there is no any intra-layer connection within both visible and hidden layer. The architecture of a typical RBM model is shown 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. Deep belief network

In general, as a deep feedforward network, DBN is built by a set of stacked RBMs, and trained by a layer-by-layer learning algorithm in an unsupervised greedy fashion. Especially, the features obtained by a RBM are viewed as the input data for a next RBM. Thereby, RBMs in a DBN are trained one by one, proceeding from the lower-level RBM and progressively shifting up in the hierarchy. In this way, DBN can increasingly capture deep features of input data. A typical DBN architecture is shown in Fig. 2, where it contains four layers: one visible input layer, three RBM hidden layers and one output layer, and the visible input layer is the input layer of the first RBM.

Generally speaking, the DBN learning consists of pre-training and fine-tuning. During the pre-training, input data is loaded to the visible input layer, and then the first RBM maps it to own

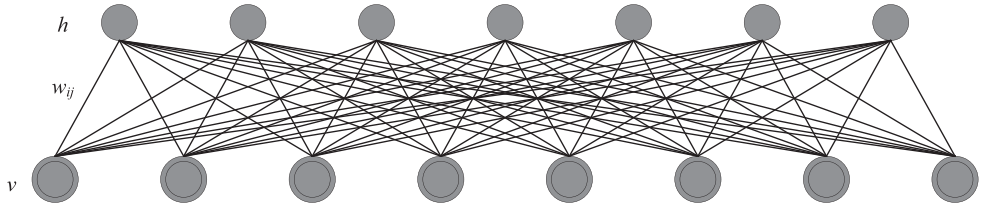


Fig. 1. An illustration of an RBM structure.

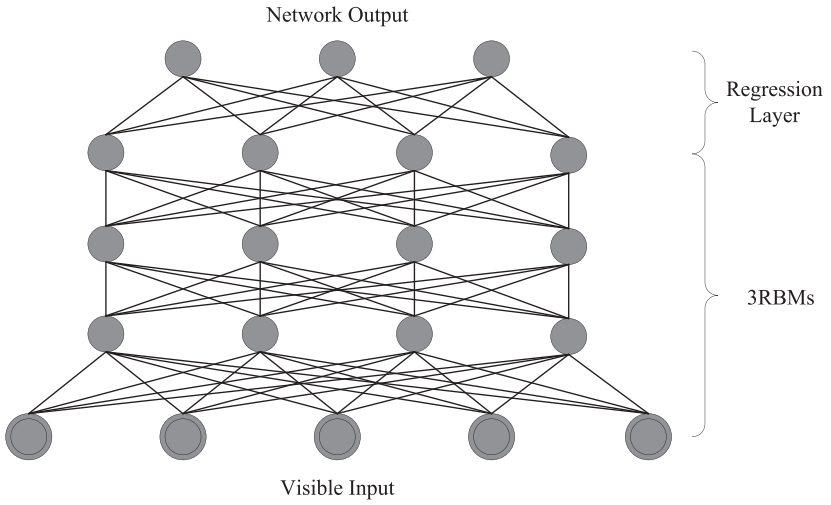


Fig. 2. An illustration of a DBN with stacked RBMs.

hidden layer for low-level feature extraction, whose the training manner is the same as the mentioned RBM. Once the current RBM training is completed, the corresponding output is viewed as the input of the next RBM for the subsequent training. This manner is repeated continuously until the desired DBN has been built. The outputs of the last RBM are learnt features over the whole pre-training process. It implies that the data have been labeled through the unsupervised learning.

In the additional logistic regression layer, the fine-tuning is performed with a training set of labeled inputs in a supervised fashion. As we all know, the popular back-propagation algorithm is used to adjust the weights of each RBM. Actually, it is a global parameter adjustment conducive to finding a minimum in a peripheral region of parameters initialized by DBN. After all RBMs are trained, DBN can be utilized to solve the classification and regression problems.

3. DSCR

To improve the nonlinear approximation capacity, we introduce the deep learning methodology into the minimum complexity ESN with a simple cycle reservoir (abbreviated as SCR in Ref. [11]) to construct a novel deep neural network architecture, termed as DSCR. Fig. 3 shows its schematic representation. Obviously, DSCR is built by replacing the onefold input layer of the original SCR structure with a pre-training input network (PIN). Intuitively, the PIN structure is a typical DBN, whereas there is a profound difference: its training is completely independent on the following ESN-based learning. In fact, PIN serves as a feature detector that is responsible for capturing the dependence in data, which offers excellent starting points for the subsequent approximation. Especially, in the proposed deep architecture, PIN becomes affiliated with SCR in a full-connection way. In theory, our proposal can guarantee a favorable approximation performance.

3.1. PIN

In structure, PIN is composed of stacked RBMs, pre-trained by a greedy layer-wise fashion. It is interesting to note that PIN itself is fixed, once its training is achieved. In other words, the PIN learning in DSCR is not influenced by the training in the next phase.

Here, consider a RBM with binary visible units $v = \{0, 1\}^D$ and hidden units $h = \{0, 1\}^F$. It should be especially mentioned that visible units use probabilities instead of sampling binary values during the RBM training, and the final update of hidden units also uses probabilities.

The energy function of the joint configuration of the units $(\boldsymbol{v}, \boldsymbol{h})$, determined by the weights and biases of RBM, is given by

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

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

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

where $Z(\theta)$ is a normalizing constant, obtained by summing over all possible energy configurations $E(\boldsymbol{v}, \boldsymbol{h}; \theta)$, that is,

$$Z(\theta) = \sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\ \boldsymbol{h};\ \theta)) \quad (3)$$

Given a training sample (visible unit), the probability of an assigned visible vector $\boldsymbol{v}$ in the RBM is calculated over the marginal distribution of $p(\boldsymbol{v}, \boldsymbol{h}; \theta)$ with regard to the space of hidden vectors.

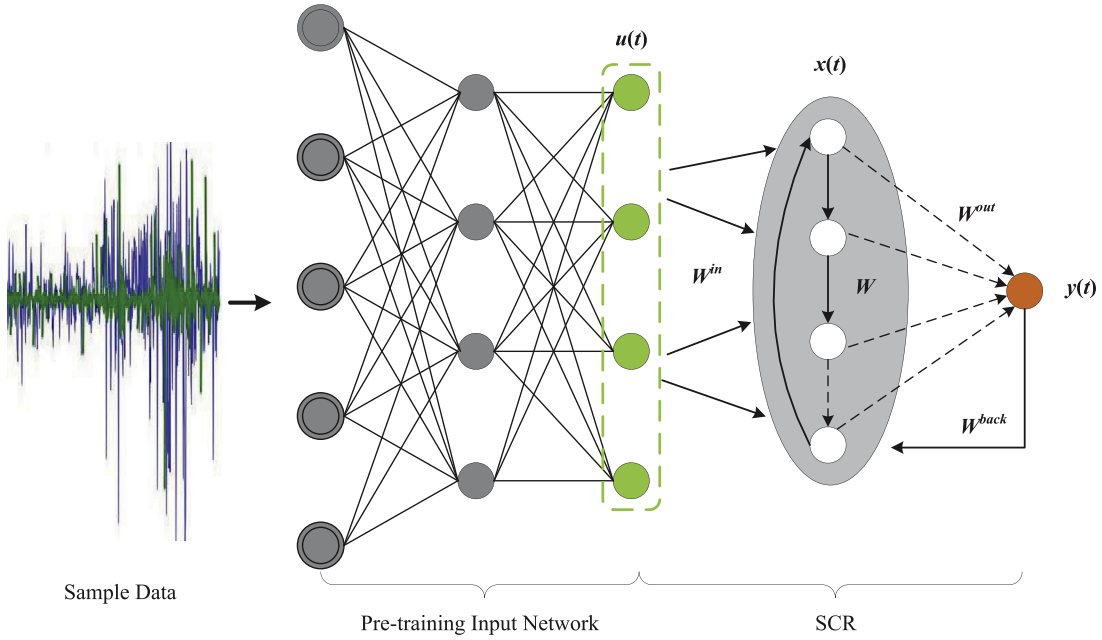


Fig. 3. Schematic representation of the DSCR architecture. (For interpretation of the references to colour in the text, the reader is referred to the web version of this article.)

$$p(\boldsymbol{v};\ \theta) = \frac{1}{Z(\theta)} \sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\ \boldsymbol{h};\ \theta)) \tag{4}$$

In order to update the weights, it is essential to compute the following derivatives of the log probability of a visible vector $\boldsymbol{v}$:

$$\frac{\partial \log p(\boldsymbol{v};\ \theta)}{\partial w_{ij}} = \langle v_i\, h_j \rangle_{data} - \langle v_i\, h_j \rangle_{model} \tag{5}$$

where $\langle \cdot \rangle$ stands for the expectation operation with respect to the distribution specified in the subscript, and the $\langle \cdot \rangle_{data}$ and $\langle \cdot \rangle_{model}$ denote the probabilities of data-driven and reconstructed-data-driven, respectively. Using the gradient of log likelihood $\log p(\boldsymbol{v};\theta)$, the updating rule for the weights is given by

$$\Delta\omega_{ij} = \eta\left(\langle v_i\, h_j \rangle_{data} - \langle v_i\, h_j \rangle_{model}\right) \tag{6}$$

where η is the learning rate. It is reasonably easy to compute the first expectation $\langle v_i\, h_j \rangle_{data}$. For the given visible vector $\boldsymbol{v}$, the conditional probability, corresponding to the hidden unit j, is given by

$$p(h_j = 1|\boldsymbol{v}) = \sigma\left(b_j + \sum_i v_i w_{ij}\right) \tag{7}$$

where the logistic sigmoid function $\sigma(\cdot)$ is defined as $\sigma(x) = 1/(1+\exp(-x))$. Therefore, we can effortlessly obtain the unbiased sample of $\langle v_i\, h_j \rangle_{data}$. Likewise, because there exists no any visible–visible connection, the conditional probability of visible unit i for the given hidden vector $\boldsymbol{h}$ is written as:

$$p(v_i = 1|\boldsymbol{h}) = \sigma\left(a_i + \sum_j h_j w_{ij}\right) \tag{8}$$

However, computing the $\langle v_i\, h_j \rangle_{model}$ is much more intractable, since the used Gibbs sampling has been demonstrated to be rather time-consuming in order to obtain unbiased samples from the model distribution. To solve this problem, Hinton et al. proposed a faster learning algorithm [24] based on contrastive divergence, which could approximate the gradient using the $\langle v_i\, h_j \rangle_{recon}$ instead of $\langle v_i\, h_j \rangle_{model}$, yielding

$$\Delta\omega_{ij} = \eta\left(\langle v_i\, h_j \rangle_{data} - \langle v_i\, h_j \rangle_{recon}\right) \tag{9}$$

$\langle v_i\, h_j \rangle_{recon}$ can be calculated by setting the visible units to a random training vector, and then the states of hidden units are obtained using Eq. (7), followed by calculating the binary states of the visible units using Eq. (8). The obtained visible states are viewed as a reconstruction from the original visible vector. Again, the states of hidden units are computed based on the reconstructed visible state using Eq. (7). Finally, the weight is updated using Eq. (9), which means that we achieve the training of RBM.

Through repeating the above process, we can achieve the training of the whole PIN. The output states of last RBM are the most representative features of input data.

3.2. SCR

SCR is used to learning the obtained features, i.e., $u(t)$, marked in green in Fig. 3. In fact, the output states of last RBM are viewed as the inputs of SCR, and then mapped to the reservoir state space to conduct the supervised training. It is essentially a simple linear regression problem with respect to the weights adjustment [12,33–36]. The corresponding reservoir state and network readout in DSCR are updated according to

$$x(t+1) = f(W^{in}u(t+1) + Wx(t) + W^{back}y(t)) \tag{10}$$

$$y(t+1) = W^{out}x(t+1) \tag{11}$$

where $x(t)$ is the reservoir state at time step t, $u(t)$ is the perceived signal injected to SCR at time step t, $y(t)$ is the obtained value of the readout at time step t, W is the internal weight matrix of connections between the internal units, W^{in} is the input weight matrix of connections between the input units and the internal units, W^{back} is the feedback weight matrix of the connections that project back from the output to the internal units, W^{out} is the output weight matrix of connections from internal units to output units, and f is the activation function of reservoir neurons (typically a hyperbolic tangent or some other Sigmoidal function).

Unlike the standard ESN, the reservoir in DSCR is organized as a cycle architecture. Nonzero elements of W lie on the sub-diagonal $W_{i+1,i} = r$ and at the upper-right corner element $W_{1,N} = r$, where N denotes the reservoir size, and r is a random value over an interval

$(-1, 1)$. In addition, all elements of W^{in} have the same absolute weigh value $v > 0$.

Essentially the supervised training in SCR is to find an optimal output weight W^{out}, which is solved by means of a simple linear regression, expressed as

$$W^{out} = M^{-1}Q \tag{12}$$

where M^{-1} is the inverse matrix of the reservoir activities over neurons and Q is a matrix of target output values. Every row of both matrices contains the reservoir states and desired output of the network at a given time step, respectively.

Except for feature extraction, DSCR also inherits outstanding inherent characteristics of the SCR architecture. Hence, the crucial working principle is still that it must possess the echo-state property (ESP): the states of the reservoir $x(n)$ should be uniquely defined by the input history. But different from the typical ESN representation, the input here refers to the visible input of whole DSCR architecture. To ensure the *ESP*, the largest singular value of W, i.e., spectral radius, must be less than 1. Moreover, due to the influence of initial reservoir states on the solution of Eq. (12), a certain number of initial steps need to be abandoned during the training, called washout phase.

3.3. DSCR learning algorithm

As discussed in Sections 3.2 and 3.1, the learning of DSCR comprises of the trainings of PIN and SCR. The former is to train the weights between visible layer and hidden layer in each RBM in the aforementioned layer-wise greedy way, while the latter is to find an optimal W^{out} through a simple linear regression.

In the first process, RBMs in PIN are pre-trained sequentially with the raw sample dataset, where the weights are updated based on the contrastive divergence (CD) method, as depicted in Algorithm 1. Especially, the visible input layer is regarded as the zero layer, and we just consider one iteration for the CD algorithm, i.e., CD-1 [15]. After initialization, the training dataset is injected into the input layer (visible input layer) of first RBM, i.e., v^0. In the positive phase, the binary states of the hidden units h^0 is computed using the v^0 and Eq. (7), and then the corresponding values are sampled: $h^0 \sim p(h^0 = 1|v^0)$ (see Lines 4–8). Next, in the negative phase, for the h^0 the values of reconstructed input units are sampled: $v^1 \sim p(v^1 = 1|h^0)$ (see Lines 9–11). After the states of the original hidden units h^0 are reconstructed again, the weights and biases can be updated (see Lines 14–17). It means that we achieve the training of the first RBM. By repeating the above process until the number of layers is up to l, the PIN training is eventually completed. What needs to be stressed is that the kth RBM, for $k = 2$ to l, is trained with the outputs of the hidden units in the $(k-1)$th layer as own inputs (see Line 6).

Algorithm 2 shows the pseudo code of the SCR training. First of all, the related parameters, such as W^{in}, W, W^{back}, are initialized and never changed during the training process (see Lines 1–3). Second, when the inputs, namely the state vectors of the last trained RBM, are loaded into the reservoir, the reservoir states are updated using Eq. (10) for $t = 0, 1, \ldots, T$. Moreover, the network states $(u(n+1), x(n+1), y(n))$ and the current outputs after washing time T_0 are collected in a state collecting matrix M of size $(T - T_0 + 1) \times (K + N + L)$ and a teacher collecting matrix Q of size $(T - T_0 + 1) \times L$ (see line 4–9), where K, N and L denote the sizes of input, reservoir and output units, respectively. Finally, the output weight matrix W^{out} is obtained by computing Eq. (12) to minimize the training error function, which implies that the SCR training is completed (see Line 11). After that, our DSCR is able to be conducted for nonlinear approximation tasks.

Algorithm 1: PIN training algorithm.

Input: X is the input vector; v, h are the vectors of output values of visible units and hidden units, respectively; l is the number of layers to train; η is the learning rate for the RBM training.

Output: w^k is weight matrix at level k, for k from 1 to l; b^k and c^k are offset vectors for the visible and hidden units at level k; $\boldsymbol{h^l}$ **is the output states of the last RBM**.

1 $v^0 \leftarrow X$;
2 **for** $k = 1$ *to* l **do**
3 initialize $w^k = 0$, $b^k = 0$, $c^k = 0$;
4 **for** *all hidden units* j **do**
5 **if** k *is not equal to 1* **then**
6 $v^{k-1} \leftarrow h_j^{k-2}$;
7 compute probability $p(h_j^{k-1} = 1|v^{k-1})$;
8 sample $h_j^{k-1} \in \{0, 1\}$ from $p(h_j^{k-1} = 1|v^{k-1})$;
9 **for** *all visible units* i **do**
10 compute probability $p(v_i^k = 1|h^{k-1})$;
11 sample $v_i^k \in \{0, 1\}$ from $p(v_i^k = 1|h^{k-1})$;
12 **for** *all hidden units* j **do**
13 compute probability $p(h_j^k = 1|v^k)$;
14 **for** *all weight values* i, j **do**
15 $w_{ij}^k \leftarrow w_{ij}^k + \eta(v_i^{k-1}h_j^{k-1} - v_i^k p(h_j^k = 1|v^k))$;
16 $b_i^k \leftarrow b_i^k + \eta(v_i^{k-1} - v_i^k)$;
17 $c_j^k \leftarrow c_j^k + \eta(h_j^{k-1} - p(h_j^k = 1|v^k))$;
18 Sample the state vector $\boldsymbol{h^l}$ of the last RBM;

Algorithm 2: SCR training algorithm.

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

Output: W^{out} is the output weight matrix.

1 ♯ **Network Initialization**
2 W^{in}, W , W^{back} are assigned according to **Section 3.2**;
3 $\lambda \in (0, 1)$, $x(0) = 0$;
4 ♯ **Sampling Network Training Dynamics**
5 $u(t) \leftarrow \boldsymbol{h^l}$;
6 **for** $t = 0$ *to* T **do**
7 update the reservoir states using the Eq. (10);
8 compute the state collecting matrix M;
9 compute the teacher collecting matrix Q using the Eq. (11);
10 ♯ **Output weights computation**
11 Solve W^{out} using the Eq. (12).

4. Experiments

To validate the effectiveness of the proposed DSCR, a great deal of simulation studies is carried out for five classical benchmark tasks: the NARMA system, the Mackey–Glass system, the multiple superimposed oscillator problem, the sunspot series and the video traffic series. The considered datasets are divided into three

Table 1
Configuration of the evaluated models in the considered experiments.

Parameter	NARMA	MG system	MSO	Sunspots	Video traffic
$CD-k$	CD-5	CD-1	CD-1	CD-1	CD-1
Batch-size α	50	10	10	10	5
Momentum β	0.05	0.55	0.6	0.6	0.05
Learning rate η	0.1	0.1	0.001	0.5	0.001
Reservoir size N	150	15	10	3	35
Spectral radius λ	0.8	0.75	0.65	0.8	0.8
Input weight v	0.6	0.8	0.4	0.9	0.7
Reservoir weight r	0.75	0.85	0.55	0.8	0.95

parts for training, hold-out validation and testing, defined by L_{trn}, L_{val}, and L_{tst} respectively. The first 100 values of training and testing sequences are discarded to wash out the initial transient. To demonstrate the advantages of our approach, we also evaluate the deep belief network (DBN) [24], the classical ESN (CESN), the minimum complex ESN architecture (SCR) [11] and LSTM. The experimental configuration is depicted in Table 1. Especially, the parameters related to LSTM is specified in [2] for the benchmark and real-world tasks. In the experiments, we use the normalized root-mean-square error (NRMSE) to measure the prediction accuracy of the evaluated models, given by

$$NRMSE = \sqrt{\frac{\sum_{t=1}^{l_{test}} (y_{test}(t) - d(t))^2}{l_{test} \cdot \sigma^2}} \tag{13}$$

where l_{test} is the length of test samples, $y_{test}(t)$ and $d(t)$ are the test output and desired output at time step t, respectively, and σ^2 is the variance of desired output $d(t)$. All the simulations are conducted 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 nonlinear autoregressive moving average (NARMA) system [11] is a classical discrete time system, and has been widely applied to test the performance of neural networks. However, it is rather difficult for modeling this system, primarily due to the nonlinearity and possibly long memory, In the experiment, we consider the following 10th order NARMA system:

$$y(n) = 0.3y(n-1) + 0.05y(n-1)\left[\sum_{i=1}^{10} y(n-1)\right] + 1.5x(n-10)x(n-1) + 0.1 \tag{14}$$

where $y(n)$ is the system output at time n, and $x(t)$ is the system input at time n, randomly drawn from a uniform distribution over the interval [0, 1]. These models are trained on the nonlinear system identification task with $L_{trn} = 6000$, $L_{val} = 3000$ and $L_{tst} = 6000$. The DSCR architecture is set to 1-40-150-1.

Fig. 4 shows the prediction results of the evaluated models over a selected region from time step 2100 to 2200 in the 10th order NARMA task. In this plot, we observe that DSCR can perform much better than the considered alternatives, being capable of fitting the target signal more accurately. In contrast, DBN is the worst performing architecture. It is entirely powerless to reconstruct the target signal. The testing NRMSEs for DSCR, DBN, SCR, CESN and LSTM are 0.0832, 1.0003, 0.4103, 0.4331, 0.3286, respectively.

To investigate the effect of deep learning parameters on prediction accuracy [25], in Fig. 5, we examine the results obtained when the different CD-k, leaning rate η and batch-size α are used. As we observe, DSCR is overwhelmingly superior to DBN over the wider choices of these parameters for the 10th order NARMA task. Especially, in Fig. 5(a), when CD-k is just up to 5, our model can obtain satisfactory performance. The smaller η is conducive to better prediction performance, while α has a slight effect on it (see Fig. 5(b) and (c)).

Finally, we evaluate the performance sensitivity on the different reservoir parameters in Fig. 6, such as reservoir size N, spectral radius λ, input weight v and reservoir weight r. In this figure, it can be seen that our DSCR achieves lowest prediction error. Noticeably, in a situation when only 150 reservoir neurons are used and $\lambda = 0.8$, it is able to offer a startling prediction accuracy, as shown in Fig. 6(a). However, SCR and CESN could never work so well, even if N and λ rise up to 500 and 0.9, respectively. As for v and r, the region where DSCR shows the best prediction accuracy lies between $v \in [0.5, 1]$ and $r \in [0.5, 1]$.

4.2. Mackey–Glass system

The Mackey–Glass (MG) system [12,33] offers a classical benchmark task for time series modeling, and its significant feature is the chaotic attractor. This system has been widely applied to test model performance for dynamical system identification in many literatures. The MG system is deduced by using a time-delay differential system with the following form

$$\frac{dy(t)}{dt} = \frac{ay(t-\tau)}{1 + y^n(t-\tau)} + by(t) \tag{15}$$

where $n = 10$, $a = 0.2$, and $b = -0.1$. If and only if delay time $\tau < 16.8$, the MG system is able to show chaotic behavior. In the experiment, we consider the MG system with $L_{trn} = 6000$, $L_{val} = 3000$ and $L_{tst} = 6000$ in the case of delay time $\tau = 17$, similar to [12], and the DSCR architecture of 1-5-5-15-1.

In Fig. 7, we show comparative diagrams of the original MG and ones obtained by DSCR, DBN, SCR, CESN and LSTM. As seen from this figure, DSCR is capable of effectively tracking the trajectory of the original MG attractor, followed by LSTM, SCR and CESN, whereas DBN is unable to reconstruct it. The corresponding testing NRMSEs are 0.00064, 1.1354, 0.2766, 0.2793 and 0.1105 respectively.

As we all know, if the time delay τ becomes larger, the MG system exhibits more severe nonlinearity. It is extremely difficult to approach such a nonlinear dynamic system using the existing models with the increase of τ. Undoubtedly, it is a serious challenge. Table 2 depicts the prediction performance of the evaluated models versus τ. Our results indicate that DSCR works considerably better than the considered alternatives. Even though τ rises to 27, which means that system nonlinearities become more severe and this problem is rather hard to deal with, it still has a surprising prediction accuracy. LSTM is the second-best performing model in the case. In addition, note that both SCR, CESN and LSTM have conspicuous degradation in prediction performance as τ increases.

Fig. 8 shows the effect of reservoir parameters on the prediction performance. We observe that for relatively small N and λ ($N \in [50, 250]$, $\lambda \in [0.1, 0.6]$), DSCR significantly outperforms SCR and CESN, while becoming slightly worse than them with the increase of both parameters (see Fig. 8(a)). We also observe that for different v and

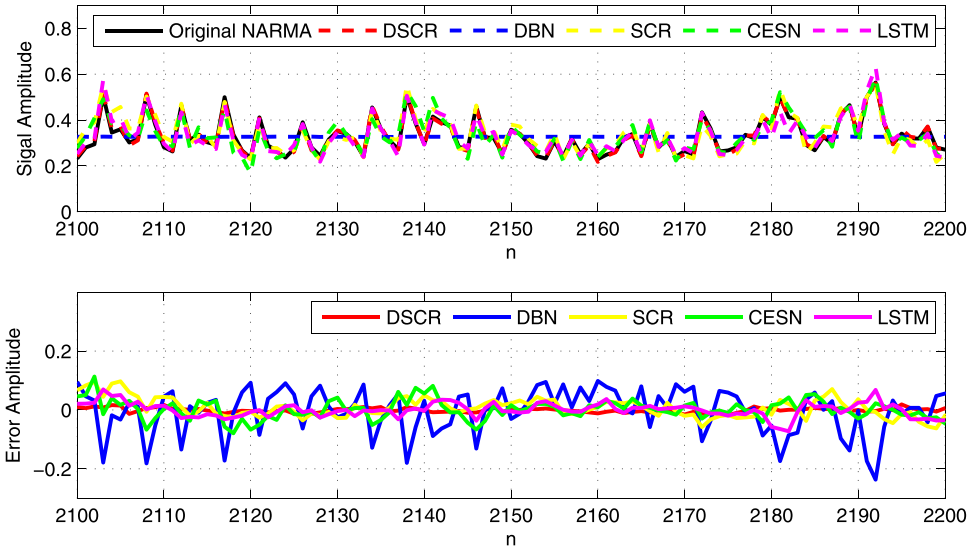
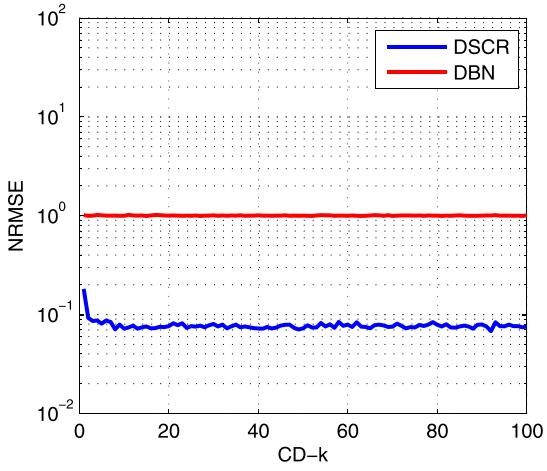
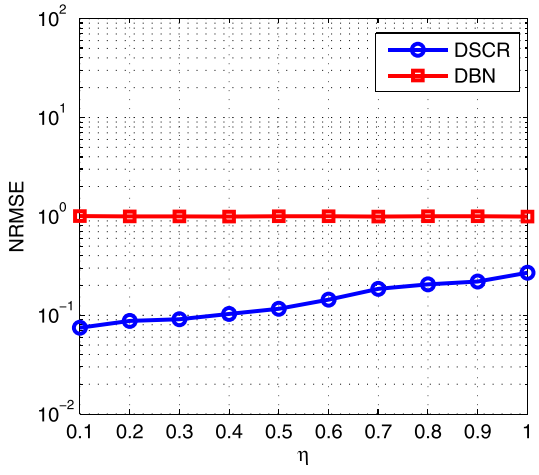


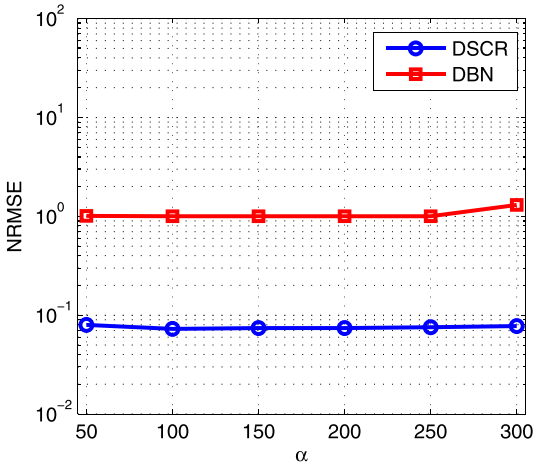
Fig. 4. Output and error of trained DSCR, SCR, CESN and DBN for the 10th order NARMA task (the 100 points of the testing sequence).



(a) CD iterations

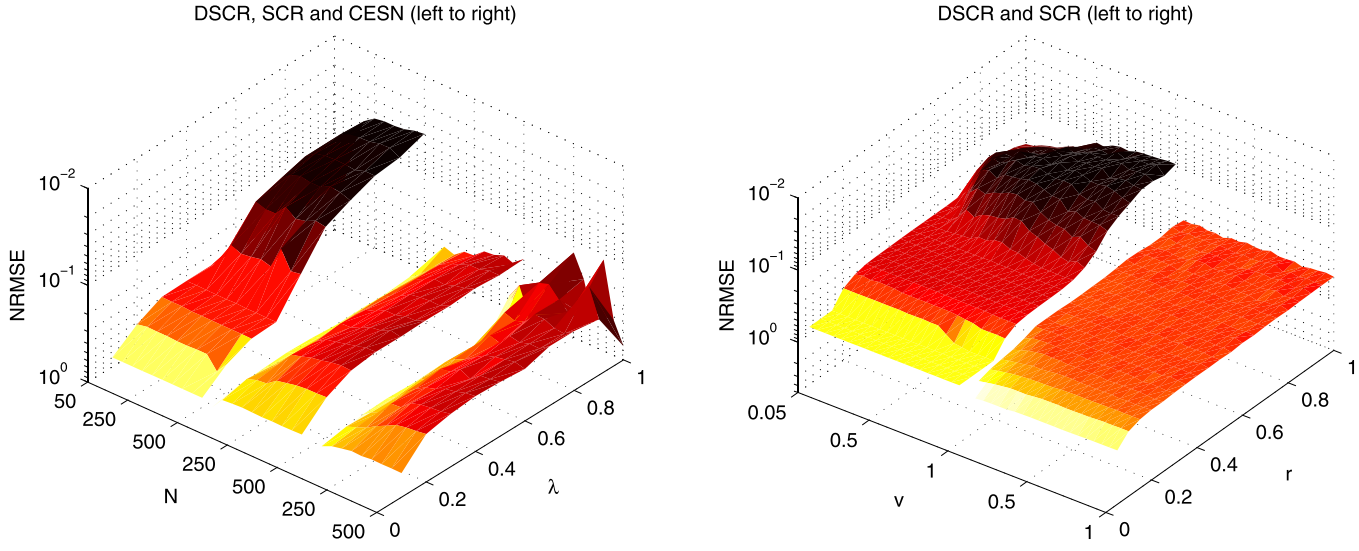


(b) Learning rate



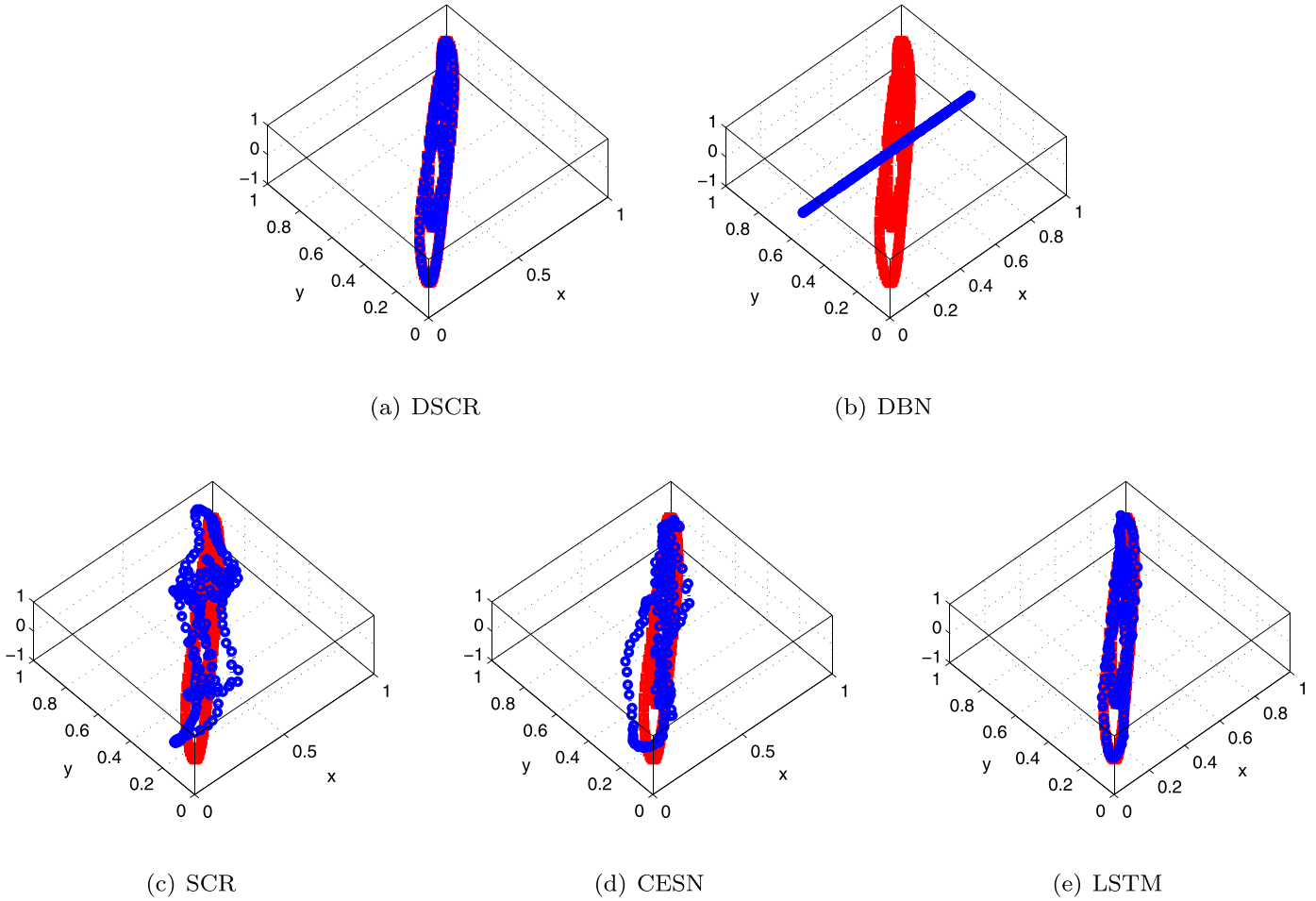
(c) Batch size

Fig. 5. Test set performance of DSCR and DBN on (a) CD iterations, (b) Learning rate η and (c) Batch size α for the 10th order NARMA task.



(a) NRMSE versus N and λ (b) NRMSE versus v and r

Fig. 6. Sensitivity of DSCR, SCR and CESN on the reservoir parameters for the 10th order NARMA task.



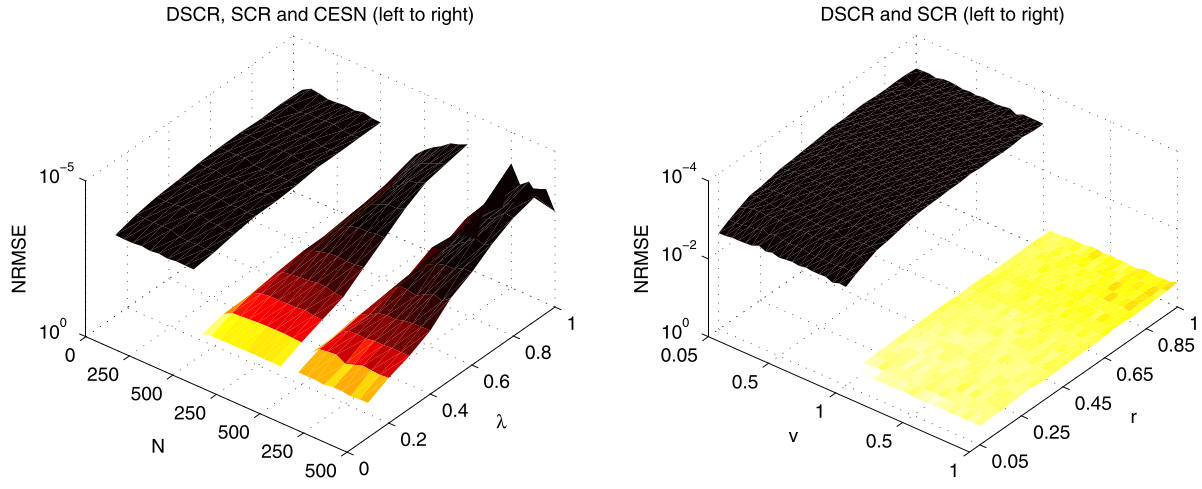
(a) DSCR (b) DBN

(c) SCR (d) CESN (e) LSTM

Fig. 7. Attractors obtained by (a) DSCR, (b) DBN, (c) SCR, (d) CESN, and (e) LSTM, where the original MG attractor is marked by red. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 2
Test set performance of DSCR, DBN, SCR, CESN and LSTM versus time delay τ for the MG system.

Model	18	19	20	21	22	23	24	25	26	27
DSCR ($\times 10^{-4}$)	6.018	6.064	6.508	6.343	5.958	5.941	5.920	5.993	5.947	6.012
DBN	0.9994	0.9967	1.0496	1.0582	1.0557	1.0169	0.9984	0.9751	0.9624	0.9941
SCR	0.2849	0.3252	0.3665	0.3851	0.3910	0.4151	0.4501	0.4546	0.4708	0.4934
CESN	0.3210	0.3313	0.3461	0.3654	0.3653	0.3690	0.3830	0.4222	0.4323	0.4575
LSTM	0.1207	0.1292	0.1354	0.1539	0.1598	0.1625	0.1811	0.2062	0.2113	0.2466



(a) NRMSE versus N and λ (b) NRMSE versus v and r

Fig. 8. Sensitivity of DSCR, SCR and CESN on the reservoir parameters for the MG system.

r, our model SCR still has the supreme prediction performance, and in the best case, its testing NRMSE is roughly three orders of magnitude better than the one obtained by SCR (see Fig. 8(b)).

4.3. Multiple superimposed oscillator problem

In this subsection, we evaluate the performance of DSCR in solving the multiple superimposed oscillator(MSO) problem [37]. The MSO dataset is generated by summing up several simple sine wave functions, expressed as follows

$$y(n) = \sum_{i=1}^{x} \sin(\alpha_i n) \tag{16}$$

where x denotes the number of sine waves, α_i denotes the frequencies of summed sine waves, and n specifies an integer index of time steps. The MSO problem with different numbers of sine waves has been considered in the literatures [12,37], where the frequencies of 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, we use $L_{trn} = 6000$, $L_{val} = 3000$ and $L_{tst} = 6000$ for the MSO sequence. The DSCR architecture is set to 2-5-10-1.

First, the MSO2, composed of two sines: $y(n) = \sin(0.2n) + \sin(0.311n)$ $n = 1, 2, \ldots,$ is used to train the considered model representatives. Fig. 9 shows comparative curves of the network output and predicted signals obtained by the evaluated models as well as the corresponding output error over a selected region from time step 1 to 900. As we could see from these figures, our model is able to offer a most accurate prediction of the target signal. The prediction NRMSEs of DSCR, DBN, SCR, CESN and LSTM are 0.000035, 0.9970, 0.6700, 0.6705 and 0.7182, respectively. Table 3 shows the performance of each structure in consideration for different MSO tasks. As expected, we see from this table that DSCR offers the most accurate prediction. Especially for the MSO3 task, the testing NRMSE obtained by DSCR only accounts for about 0.01% of those obtained by SCR, CESN and LSTM, and 0.005% of that obtained by DBN. It indicates that our model has a remarkable capacity of solving the MSO problem well.

Table 3
Test set performance of DSCR, DBN, SCR, CESN and LSTM for the different MSO tasks.

Model	MSO3	MSO4	MSO5	MSO6	MSO7	MSO8
DSCR($\times 10^{-4}$)	0.569	0.6232	1.603	2.9238	4.7429	8.0599
DBN	0.9672	0.9484	0.9524	0.9404	0.9345	0.9111
SCR	0.5366	0.4588	0.4997	0.4432	0.3915	0.3792
CESN	0.5377	0.4579	0.4911	0.4390	0.3879	0.3587
LSTM	0.5862	0.5604	0.5375	0.5048	0.4619	0.4257

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 DSCR 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 [12], which contain 3174 sunspots numbers from January 1749 to November 2014, respectively, with $L_{trn} = 3000$, $L_{val} = 1500$ and $L_{tst} = 3000$. The architectures of DSCR is set to 1-3-3-1.

Fig. 10 shows prediction outputs and errors obtained by DSCR 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

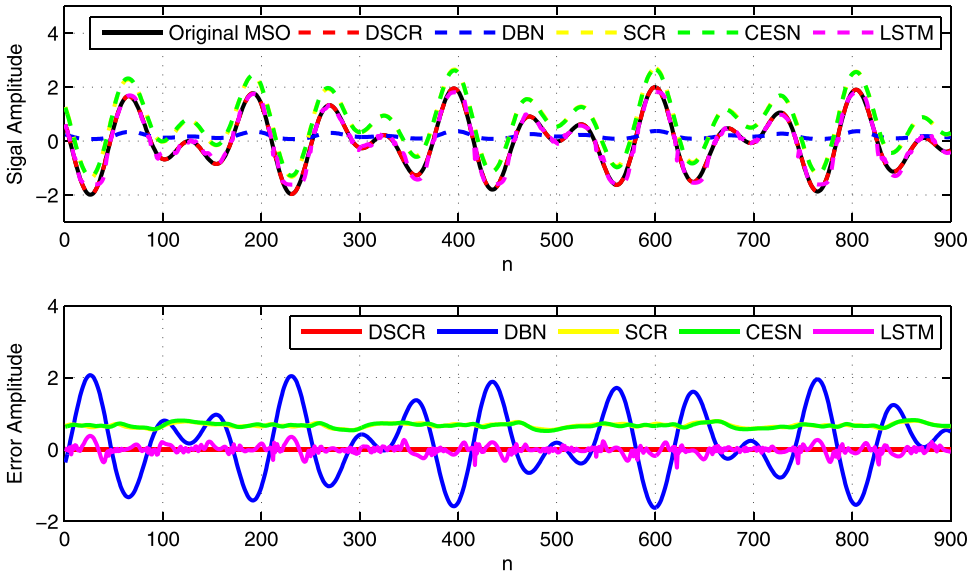


Fig. 9. Output and error of DSCR, DBN, SCR, CESN and LSTM for the MSO task.

Table 4
Testing set performance of DSCR, DBN, SCR, CESN and LSTM for the sunspots series tasks.

Data	Model	NRMSE	N	λ
Jerky sunspots series	DSCR	0.3419	3	0.8
	DBN	1.6372	-	-
	SCR	0.8104	120	0.1
	CESN	0.3571	70	0.3
	LSTM	0.4327	-	-
Smoothed sunspots series	DSCR	0.0256	3	0.8
	DBN	1.7066	-	-
	SCR	0.7470	120	0.08
	CESN	0.0298	190	0.6
	LSTM	0.0406	-	-

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 4, where the comparative models offer optimal prediction performance for the given reservoir size and spectral radius. As we observe, DSCR obtains the lowest testing NRMSE using the smallest reservoir size, while the others can hardly achieve such good prediction performance.

4.5. VBR video traffic prediction

As we all know, variable-bit-rate (VBR) video traffic prediction is crucial for efficient resource management and reliable video transmission [38–40]. Therefore, for the last task, our DSCR is tested on VBR video traces from the video trace library of Arizona State University [41], including NBC News, Star Wars IV, Tokyo Olympics, and Sony Demo. These considered video traces cover different types of content. NBC News represents a news show with frequent scene changes, Star Wars IV is derived from the motion pictures, Tokyo Olympics is a documentary related to the Olympic games, and Sony Demo is generated from demo material for a Sony high definition camera. Especially, the video datasets are equipped with a G16B3 GOP structure of [IBBBPBBBPBBBPBBB], which indicates that this structure has 16 frames with three B frames between I and P frames. Their statistical characteristics are available at http://www.trace.eas.asu.edu. In this task, we use $L_{trn} = 1000$, $L_{val} = 1500$ and $L_{tst} = 1000$, and consider the DSCR architecture of 1-15-15-35-1.

Table 5
Test set performance of DSCR, DBN, SCR, CESN and LSTM on News, Star Wars, Tokyo Olympics, and Sony Demo.

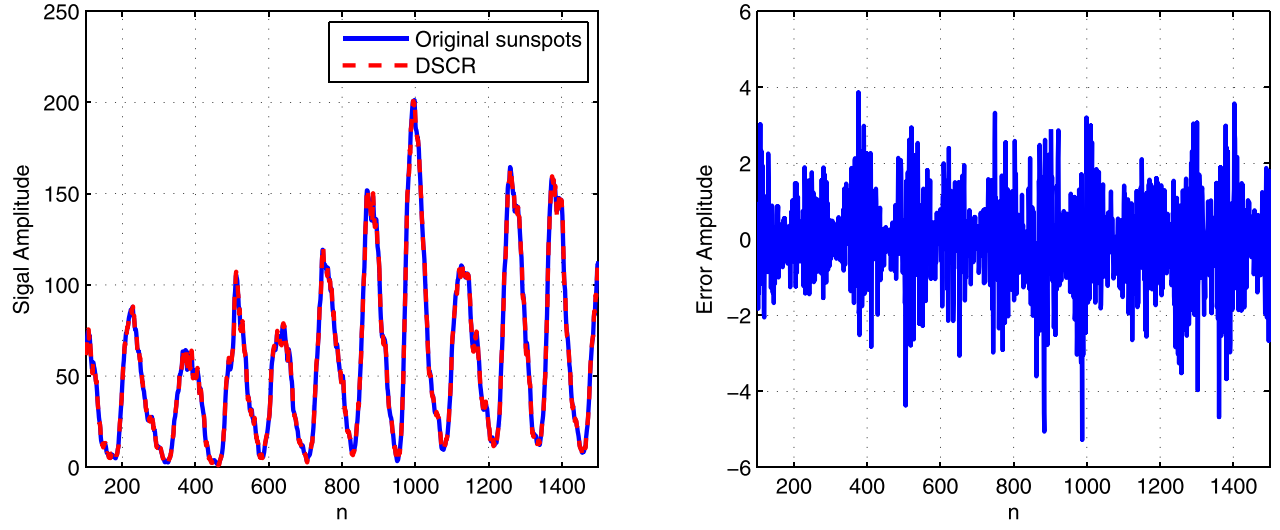
Model	News	Star Wars	Tokyo Olympics	Sony Demo
DSCR	0.4542	0.3824	0.3444	0.3567
DBN	1.1089	1.0228	1.4153	1.1883
SCR	1.0160	1.1673	1.0368	1.3373
CESN	0.9637	0.9052	0.7834	0.8913
LSTM	0.9841	0.1005	0.9972	1.1049

In Fig. 11, we show prediction results over a selected region from frame number 300 to 500 obtained by DSCR. As we could see from this figure, DSCR is able to effectively follow the original traffic, even if the abrupt variations occur in the video traces. The outstanding prediction performance of DSCR over DBN, SCR, CESN and LSTM is further depicted in Table 5.

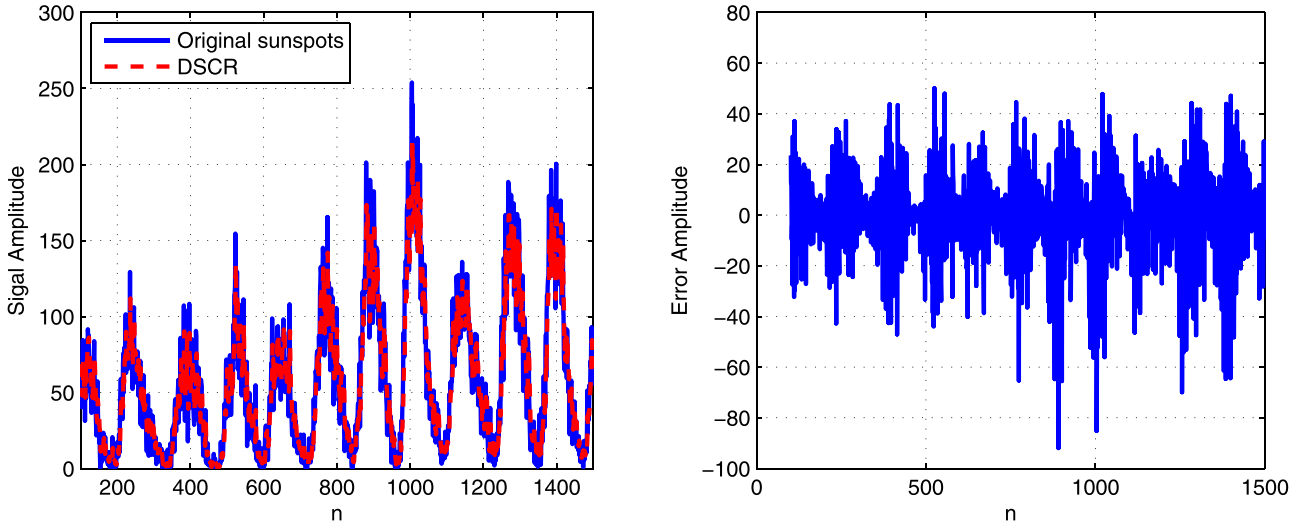
Besides, many studies have demonstrated that network traffic behaves statistical self-similarity. Generally, the Hurst parameter $H \in (0.5, 1)$ is the unique measure of the self-similarity degree, and the larger H value means the stronger degree of self-similarity. In the experiment, the popular R/S method [42] is used to evaluate the Hurst parameter. Figs. 12, 13, 14 15 show comparative self-similarity diagrams of the original and predicted video traffic, obtained by plotting the pox diagrams of R/S, for News, Star Wars, Tokyo Olympics and Sony Demo, respectively. It is clear that the self-similarity characteristic of the original video traffic is maintained in the test as much as possible, since there exists a slight change of the H value before and after prediction for each video trace. In other words, our model effectively captures the self-similarity characteristic of video traffic. Thus, from the characteristic analysis of video traffic, we again confirm that DSCR possesses superior nonlinear approximation capacity.

5. Short-term memory capacity

In order to study the temporal processing ability of the proposed DSCR, we quantified its inherent capacity that the past input information can be encoded in the instantaneous spatial state of the system, termed memory capacity (MC). It measures the ability of recovering the input signal from a past time t, as illustrated



(a) Jerky sunspots



(b) Smoothed sunspots

Fig. 10. Output and error of DSCR on the sunspots time series.

in [43] by Jaeger. For the sake of argument, we assume that the network is actuated by a univariate stationary input signal $u(t)$. For a given delay k, consider the task of outputting $s(t-k)$ after seeing the *i.d.d.* input stream $\ldots s(t-2)s(t-1)s(t)$ up to time t for DSCR. The well-fitting characteristic is measured in terms of the squared correlation coefficient between the desired output (i.e., visible input delayed by k time steps) and the observed network output $y(t)$

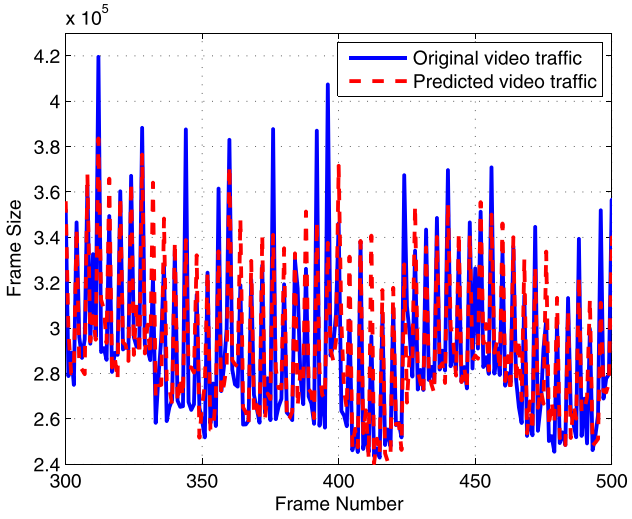
$$MC_k = \frac{Cov^2(s(t-k), y(t))}{Var(s(t))Var(y(t))} \tag{17}$$

where *Cov* denotes the covariance, and *Var* denotes variance. The short-term memory (*STM*) capacity is then formulated as:

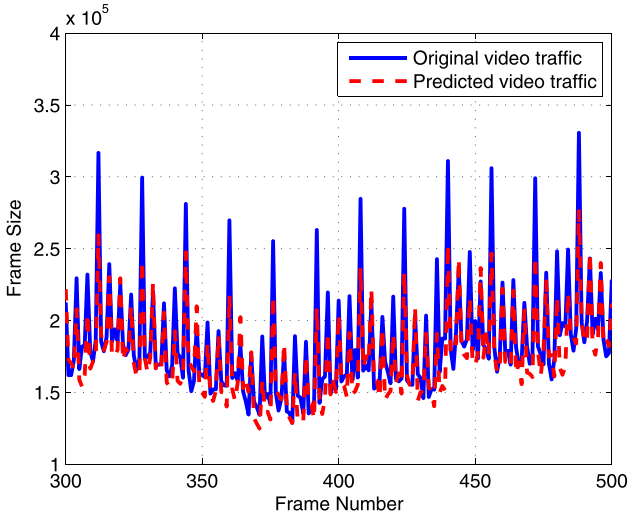
$$MC = \sum_{k=1}^{\infty} MC_k \tag{18}$$

We empirically evaluate the short-term *MC* of DSCR, SCR and CESN, where the networks are trained to memorize the inputs delayed by $k = 1, 2, \ldots, 40$. Here, 2000 samples of the 10th order NARMA dataset in Section 4.1 are used to evaluate the MC of the DSCR model. The network structure of DSCR is set to 1-5-20-40, whose parameter configuration is depicted in Table 1 except for $N = 20$.

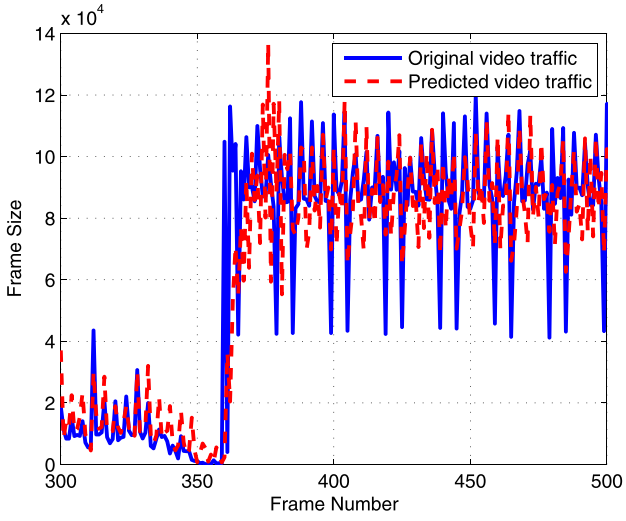
Fig. 16 shows the forgetting curves of each model versus different spectral radius λ, i.e., plots of the k-delay memory capacities, where *detCoeff* is the squared correlation coefficient (i.e., MC_k in Eq. (17)). Intuitively, DSCR has more powerful k-delay memory capacities than SCR and CESN, especially for $\lambda = 0.9$. In this case, it 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. Table 2 further depicts the STM capacities of the evaluated models, where we see that DSCR has a most excellent STM capacity. Moreover, the capacity depends heavily on spectral radius, and as λ increases, DSCR tends to possess a better memory. In order to measure the nonlinear approximation capability of the evaluated models, we average the NRMSEs across 40 reconstructions, as shown in Table 7. The experimental results again demonstrate its



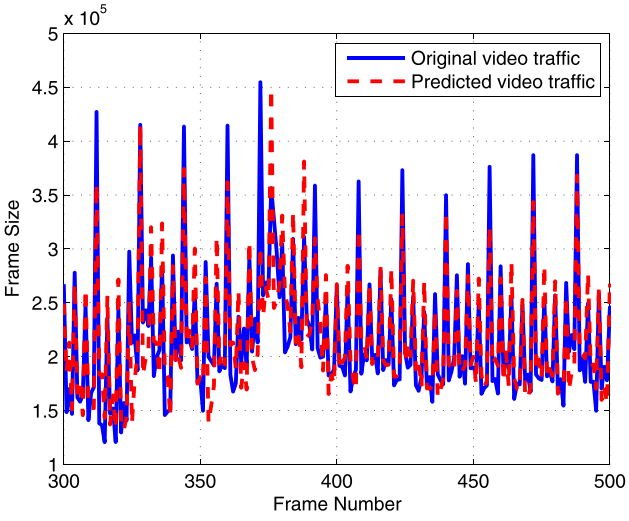
(a) News



(b) Star Wars

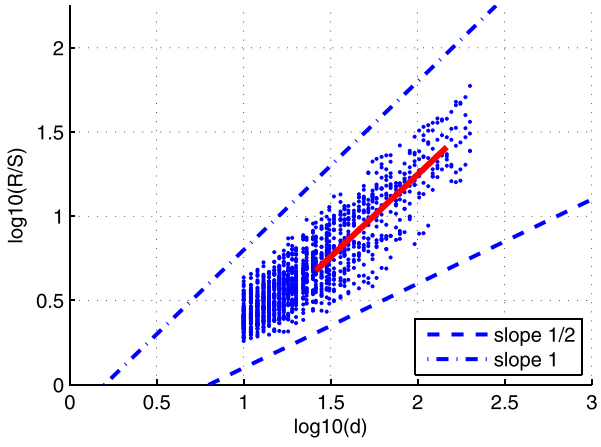


(c) Tokyo Olympics

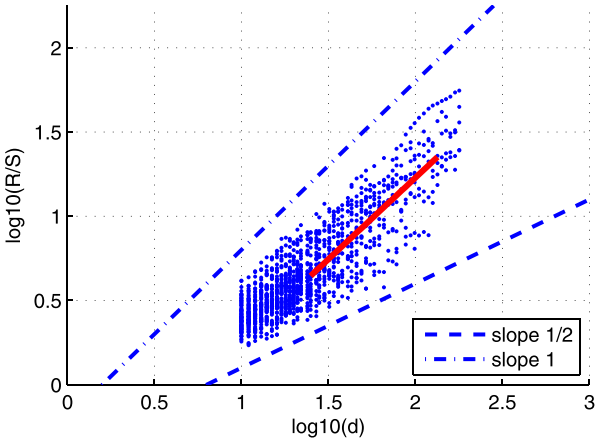


(d) Sony Demo

Fig. 11. Output of DSCR on the video traffic task, (a) News, (b) Star Wars, (c) Tokyo Olympics, and (d)Sony Demo.

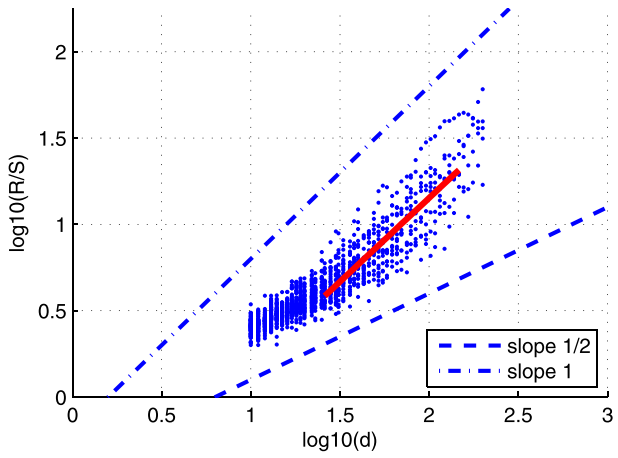


(a) Original News traffic

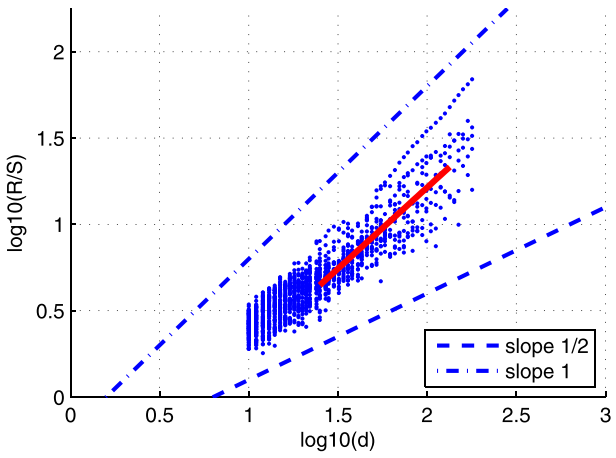


(b) Predicted News traffic

Fig. 12. Pox plots of R/S on News for DSCR. The slope of the red line is the *H* value. The lower and upper reference lines correspond to slopes of 0.5 and 1, respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

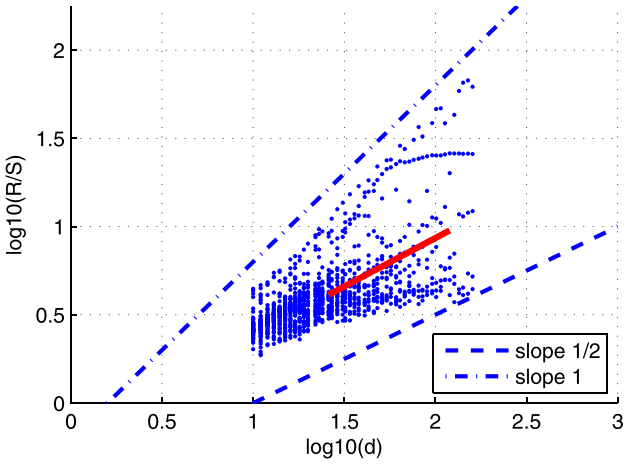


(a) Original Star Wars traffic

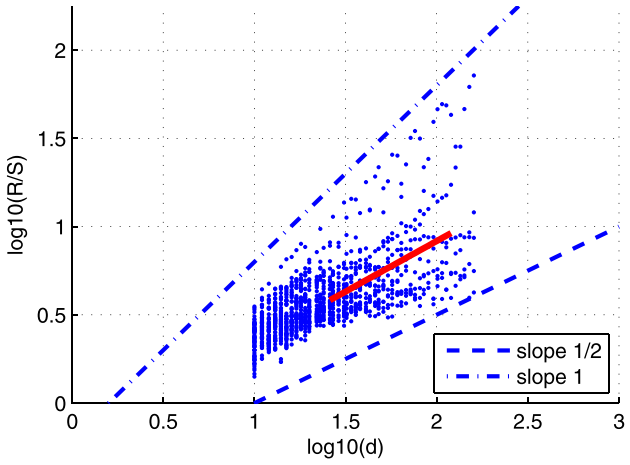


(b) Predicted Star Wars traffic

Fig. 13. Pox plots of R/S on Star Wars for DSCR. The slope of the red line is the *H* value. The lower and upper reference lines correspond to slopes of 0.5 and 1, respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

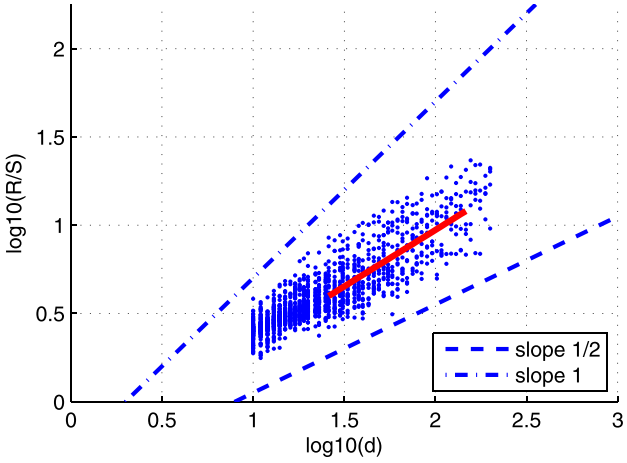


(a) Original Tokyo Olympics traffic

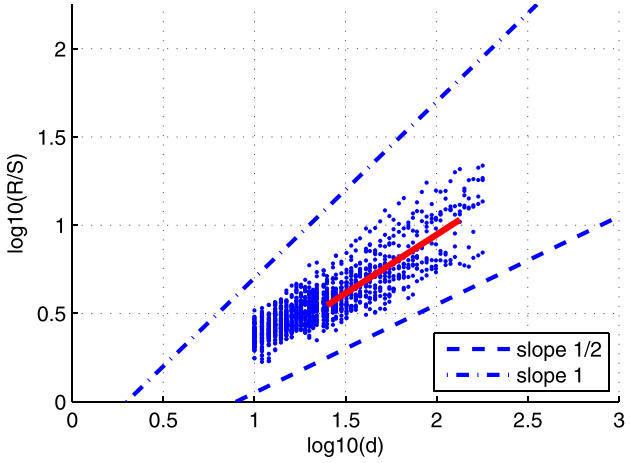


(b) Predicted video traffic

Fig. 14. Pox plots of R/S on Tokyo Olympics for DSCR. The slope of the red line is the *H* value. The lower and upper reference lines correspond to slopes of 0.5 and 1, respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

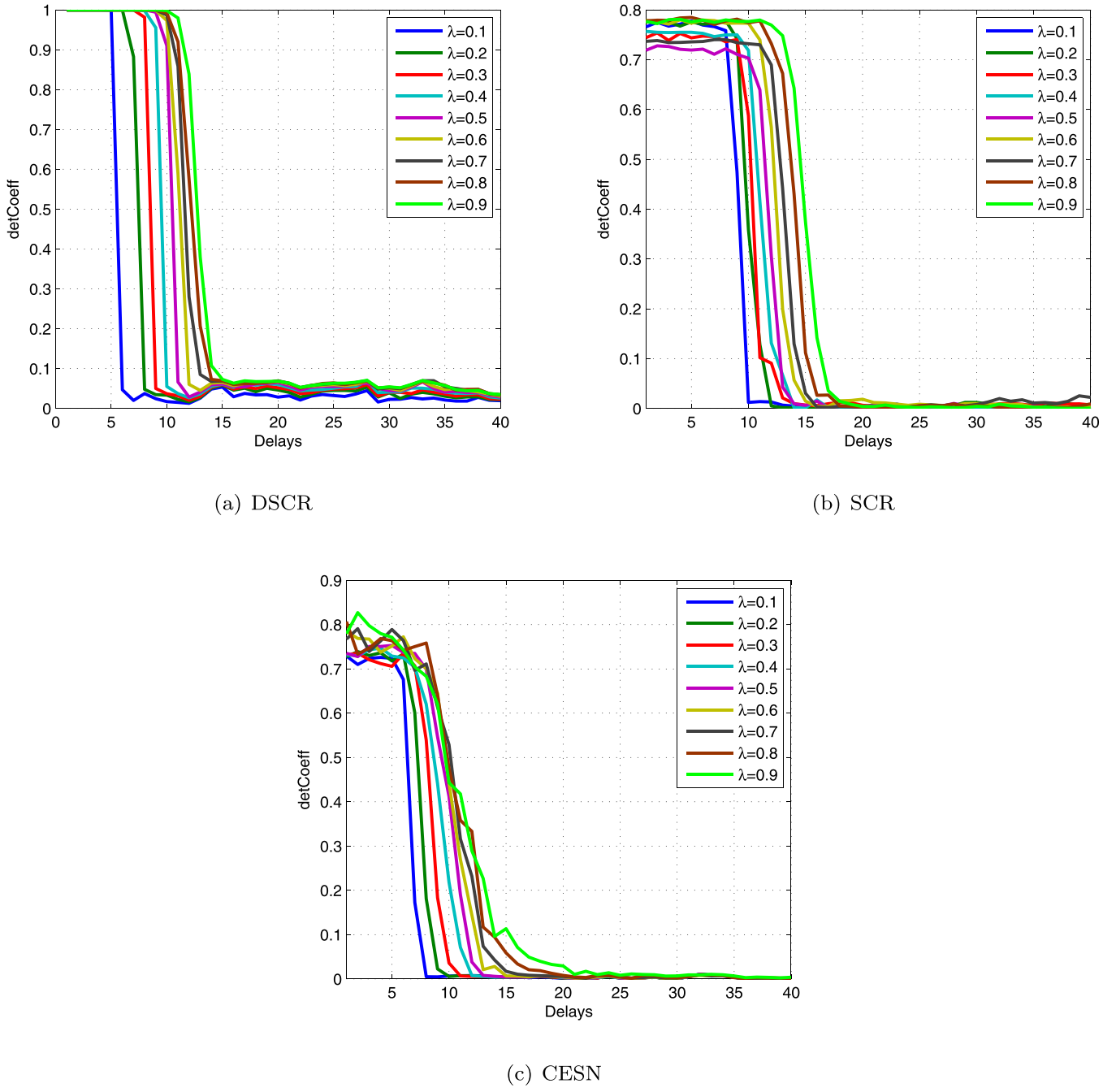


(a) Original Sony Demo traffic



(b) Predicted video traffic

Fig. 15. Pox plots of R/S on Sony Demo for DSCR. The slope of the red line is the *H* value. The lower and upper reference lines correspond to slopes of 0.5 and 1, respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)



(a) DSCR (b) SCR

(c) CESN

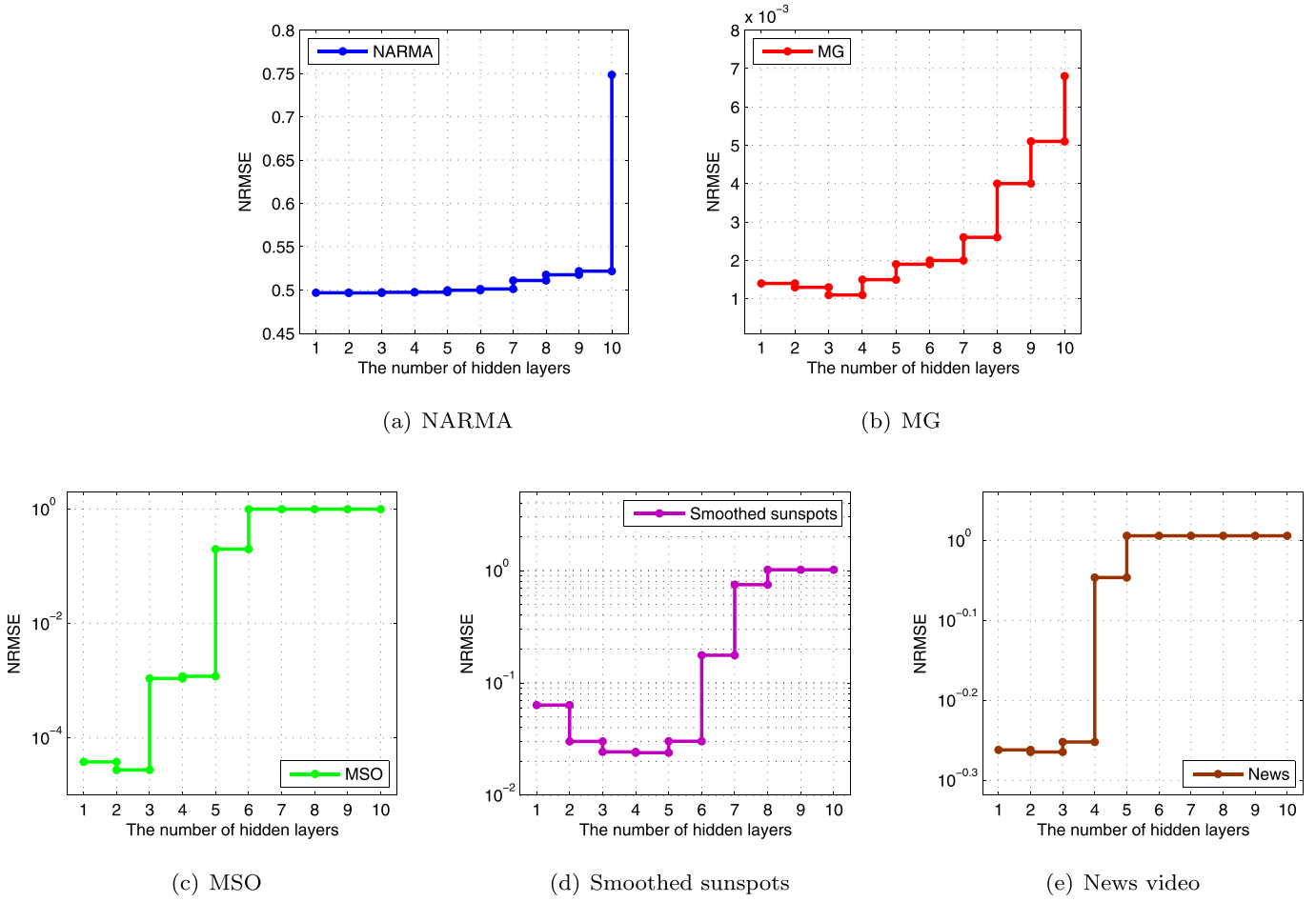
Fig. 16. The forgetting curves of (a) DSCR, (b) SCR and (c) CESN for different λ in the 10th order NARMA task. The spectral radius varies from 0.1 to 0.9.

Table 6
STM capacities of DSCR, SCR and CESN versus spectral radius λ for the 10th order NARMA task.

Model	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
DSCR	5.9874	8.1058	9.3026	10.4184	11.4353	12.1565	12.7798	13.2621	13.8246
SCR	6.7625	7.5724	7.6926	8.1571	8.3005	9.6189	9.6051	10.6474	11.3493
CESN	4.5865	5.3021	5.9115	6.5594	7.1835	7.6579	7.9619	8.3364	8.6781

Table 7
Reconstruction performance of DSCR, SCR and CESN versus the spectral radius λ for the 10th order NARMA task.

Model	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
DSCR	0.8621	0.8148	0.7858	0.7616	0.7389	0.7241	0.7107	0.6998	0.6863
SCR	0.9223	0.9082	0.9107	0.9103	0.9113	0.8712	0.8818	0.8541	0.8426
CESN	0.9755	0.9631	0.9533	0.9405	0.9271	0.9153	0.9080	0.8991	0.8863



(a) NARMA (b) MG

(c) MSO (d) Smoothed sunspots (e) News video

Fig. 17. Test set performance of DSCR on the network depth for the considered tasks, (a) NARMA, (b) MG, (c) MSO, (d) Smoothed sunspots and (e) News video.

superior ability of DSCR to recover the network inputs with high performance. For example, for $\lambda = 0.9$, our model increases the reconstruction performance to around 69%, which is 15% and 20% higher than those obtained with SCR and CESN, respectively. Interestingly enough, combined with Table 6, it can be seen that the more powerful the STM capacity is, the better nonlinear approximation capacity DSCR has.

6. Discussion

In this study, we investigate the utility of a novel recurrent neural system with minimum complexity in time series prediction. To our knowledge, this is the first report incorporating deep learning methodology into ESN modeling to address this type of problem. Experimental results over the datasets allow us to draw some important conclusions.

(1) **DSCR has much better prediction performance than the shallow models (i.e., SCR, CESN and LSTM) and the deep hierarchical model (i.e., DBN)**. The PIN structure in DSCR is capable of capturing information with different degrees of abstraction, thus allowing the model to learn more complex feature representations of the observed data. In fact, the independent PIN learning can offer excellent starting points for the following SCR training. As a result, DSCR obtains a significant enhancement of nonlinear approximation capacity, as shown in Figs. 4–11 and Tables 2–5. However, there is one notable exception that the SCR and CESN architectures can slightly outperform DSCR for the MG task as the reservoir size increases (see Fig. 8), our model is still a promising alternative to the ESN paradigm, because it is capable of achieving a superior prediction performance using a small-scale reservoir. Especially in sunspots series prediction task, DSCR, just using three reservoir neurons, effortlessly beats the considered competitors, as listed in Table 4. Furthermore, to gain a better insight into the powerful nonlinear approximation capacity, we evaluate the effect of network depth on the prediction performance in Fig. 17, where the experimental configuration is the same as Table 1 and the size of each hidden layer is set to 10. In this plot, we observe that two hidden layers in DSCR is more beneficial to the prediction accuracy for the NARMA, MSO and News video tasks, and three hidden layers is an optimal choice for the MG task, while for sunspots prediction task, DSCR needs a network depth of four layers to achieve the best approximation performance.

(2) **DSCR gives an excellent balance between learning speed and modeling performance of time series data.** To evaluate the learning speed, we give the training time of each model for the selected tasks in Sections 4.1–4.5, as shown in Fig. 18. We observe that DSCR has a rather lower training time than DBN and LSTM, resulting in a faster learning speed, since the training is achieved based on a simple linear regression method. Whereas, our model cannot perform as well as SCR and CESN in learning speed, due to the additional pre-training of the stacked RBMs. For example, for the

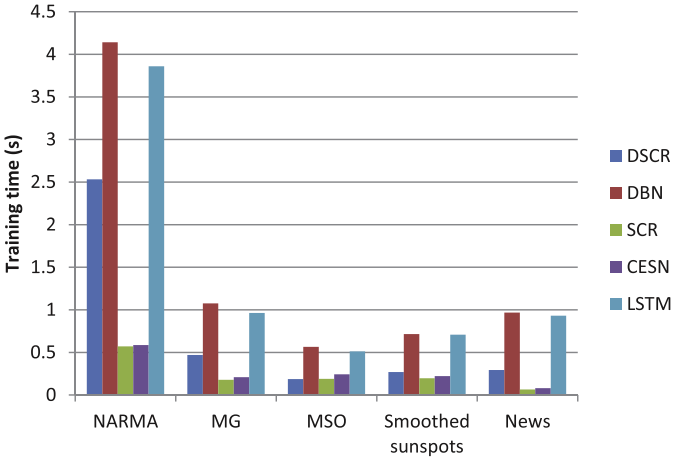


Fig. 18. The training time of the evaluated models.

10th order NARMA task, DSCR requires much more training time than the considered ESN-based models. Even so, it is still an effective model for time series prediction, because SCR and CESN can never achieve the outstanding prediction performance like DSCR (see Fig. 6), in spite of extending the reservoir size N to 500 and 400, respectively. More importantly, their corresponding training time increases dramatically up to about 8.3 s and 5.2 s. This means that our model has a fairly fast learning speed while ensuring the optimal prediction performance. In other words, it offers a very good tradeoff between computational complexity and nonlinear approximation capacity.

(3) **DSCR possesses a powerful STM capacity**. It is mainly attributed to the multi-layered RBMs in the PIN structure toward ESN that effectively capture data features, which further enhance the STM capacity on the basis of the reservoir. As a novel ESN paradigm, the nonlinear approximation capability of DSCR is directly related to its memory capacity, or more exactly, on the massive STM [44], when modeling the highly nonlinear systems. It has been proved in [43] that a larger spectral radius results in a slower decay of the networks response to an impulse input, but a stronger networks memory capacity. In the case, DSCR possesses a superior nonlinear approximation performance in theory. Fig. 16 and Tables 6–7 again verify the findings.

(4) **The deep learning and reservoir parameters have a great impact on the model performance.** In our experiments, we primarily focused on the following deep learning parameters [25]: CD iterations, learning rate and batch-size. In general, a small number of iterations on CD can minimize the reconstruction error (see Fig. 5(a)), while too many iterations lead to heavy computational burden. When the learning rate is much too large, the reconstruction error increases dramatically and the weights may explode, which leads to the significant performance degradation (see Fig. 5(b)). The reasonable batch-size can reduce the sampling error when evaluating the gradient for the whole training set from a single batch, and is usually set to 10, as shown in Fig. 5(c). Regarding the reservoir parameters, we would like to make two substantial comments. The first one concerns the reservoir size. As we notice in Figs. 6(a) and 8(a), large-scale reservoir is not beneficial to performance improvement for DSCR. This phenomenon can be explained by that its hierarchical architecture is sufficient to capture multi-level features in the data without too much reservoir neurons. Our second comment regards the performance sensitivity on the input and reservoir weights. According to Figs. 6(b) and 8(b), it is worthwhile to point out that the bigger input and reservoir weights should be chosen for given tasks to obtain superior approximation performance. In our experiments, all parameters are empirically selected for a given prediction task. As we all know, determining so many parameters is a challenging task [45,46]. Essentially, this is a multi-objective optimization problem, but beyond the scope of this paper.

6. Conclusion and further work

In the paper, we propose a deep architecture on the basis of the minimum complexity ESN with a simple cycle reservoir, namely, DSCR. This architecture employs the PIN consisting of stacked RBMs that allows our model to learn a good representation of features in the input data. The learnt features are able to offer excellent starting points for the subsequent ESN-based network training, which enables the performance improvement. Experiments on five time-series benchmarks indicate that DSCR significantly outperforms DBN, SCR, CESN and LSTM in terms of prediction accuracy, STM, and training time. However, ESN is far from explored in a deep learning framework yet. We wish this paper can motivate more studies that contribute to the performance enhancement of the ESN model. Future research will work on optimizing multiple parameters related to DSCR with optimization algorithms (e.g., particle swarm optimization and genetic algorithms).

Acknowledgments

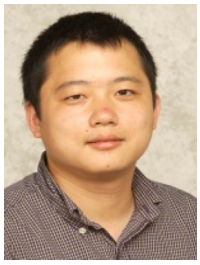
This work was funded in part by National Natural Science Foundation of China (NSFC) under grant number 61503120 and 91646116, and Science and Technology Support Program of Jiangsu Province (no. BE2016776).

References

[1] F.J. Ordóñez, D. Roggen, Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition, Sensors 16 (1) (2016) 115.
[2] F.M. Bianchi, E. Maiorino, M.C. Kampffmeyer, A. Rizzi, R. Jenssen, An overview and comparative analysis of recurrent neural networks for short term load forecasting, ArXiv preprint arXiv:1705.04378.
[3] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[4] J. Schmidhuber, D. Wierstra, M. Gagliolo, F. Gomez, Training recurrent networks by Evolino, Neural Comput. 19 (3) (2007) 757–779.
[5] H. Jaeger, The Echo State Approach to Analysing and Training Recurrent Neural Networks—With an Erratum Note, German National Research Center for Information Technology, Bonn, Germany, 2001, p. 34. GMD Technical Report 148.
[6] H. Jaeger, H. Haas, Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication, Science 304 (5667) (2004) 78–80.
[7] H. Jaeger, Adaptive nonlinear system identification with echo state networks, in: Advances in Neural Information Processing Systems, 2002, pp. 593–600.
[8] M.C. Ozturk, J.C. Principe, An associative memory readout for ESNS with applications to dynamical pattern recognition, Neural Networks 20 (3) (2007) 377–390.
[9] M. Salmen, P.G. Ploger, Echo state networks used for motor control, Proceedings of the 2005 IEEE International Conference on Robotics and Automation, 2005. ICRA 2005, IEEE, 2005, pp. 1953–1958.
[10] K. Bush, C. Anderson, Modeling reward functions for incomplete state representations via echo state networks, 2005 IEEE International Joint Conference on Neural Networks, 2005. IJCNN'05. Proceedings, Vol. 5, IEEE, 2005, pp. 2995–3000.
[11] A. Rodan, P. Tino, Minimum complexity echo state network, IEEE Trans. Neural Networks 22 (1) (2011) 131–144.
[12] J. Qiao, F. Li, H. Han, W. Li, Growing echo-state network with multiple subreservoirs, IEEE Trans. Neural Networks Learn. Syst. 28 (2) (2017) 391–404.
[13] G. Holzmann, H. Hauser, Echo state networks with filter neurons and a delay & sum readout, Neural Networks 23 (2) (2010) 244–256.
[14] D. Li, M. Han, J. Wang, Chaotic time series prediction based on a novel robust echo state network, IEEE Trans. Neural Networks Learn. Syst. 23 (5) (2012) 787–799.
[15] Y. Bengio, Learning deep architectures for AI, Found. Trends Mach. Learn. 2 (1) (2009) 1–127.
[16] N.L. Roux, Y. Bengio, Deep belief networks are compact universal approximators, Neural Comput. 22 (8) (2010) 2192–2207.
[17] O. Delalleau, Y. Bengio, Shallow vs. Deep Sum-product Networks, 2011.

[18] R. Sarikaya, G.E. Hinton, A. Deoras, Application of deep belief networks for natural language understanding, IEEE/ACM Trans. Audio Speech Lang. Process. 22 (4) (2014) 778–784.
[19] R. Salakhutdinov, G. Hinton, An efficient learning procedure for deep Boltzmann machines, Neural Comput. 24 (8) (2012) 1967–2006.
[20] X. Peng, H. Tang, L. Zhang, Z. Yi, S. Xiao, A unified framework for representation-based subspace clustering of out-of-sample and large-scale data, IEEE Trans. Neural Networks Learn. Syst. 27 (12) (2016) 2499.
[21] X. Peng, Z. Yu, Y. Zhang, H. Tang, Constructing the l2-graph for robust subspace learning and subspace clustering, IEEE Trans. Cybern. 47 (4) (2016) 1053–1066.
[22] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, et al., Greedy layer-wise training of deep networks, Adv. Neural Inf. Process. Syst. 19 (2007) 153.
[23] B. Schlkopf, J. Platt, T. Hofmann, Efficient learning of sparse representations with an energy-based model, Advances in Neural Information Processing Systems, 2006, pp. 1137–1144.
[24] G.E. Hinton, S. Osindero, Y.W. Teh, A fast learning algorithm for deep belief nets, Neural Comput. 18 (7) (2006) 1527–1554.
[25] G. Hinton, A practical guide to training restricted Boltzmann machines, Momentum 9 (1) (2010) 1–21.
[26] M.M.R. Sazal, S.K. Biswas, M.F. Amin, K. Murase, Bangla handwritten character recognition using deep belief network, 2013 International Conference on Electrical Information and Communication Technology (EICT), IEEE, 2014, pp. 1–5.
[27] A. Mohamed, G.E. Dahl, G. Hinton, Acoustic modeling using deep belief networks, IEEE Trans. Audio Speech Lang. Process. 20 (1) (2012) 14–22.
[28] Q.V. Le, W.Y. Zou, S.Y. Yeung, A.Y. Ng, Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis, Computer Vision and Pattern Recognition, 2011, pp. 3361–3368.
[29] Y. Yang, C. Deng, S. Gao, W. Liu, D. Tao, X. Gao, Discriminative multi-instance multi-task learning for 3D action recognition, IEEE Trans. Multimedia 19 (3) (2017) 519–529.
[30] Y. Chen, X. Zhao, X. Jia, Spectral spatial classification of hyperspectral data based on deep belief network, IEEE J. Select. Top. Appl. Earth Observ. Remote Sens. 8 (6) (2015) 1–12.
[31] C. Deng, J. Xu, K. Zhang, D. Tao, X. Gao, X. Li, Similarity constraints-based structured output regression machine: an approach to image super-resolution, IEEE Trans. Neural Networks Learn Syst. 27 (12) (2015) 2472–2485.
[32] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.
[33] S.P. Chatzis, Y. Demiris, Echo state Gaussian process, IEEE Trans. Neural Networks 22 (9) (2011) 1435–1445.
[34] B. Zhang, D.J. Miller, Y. Wang, Nonlinear system modeling with random matrices: echo state networks revisited, IEEE Trans. Neural Networks Learn. Syst. 23 (1) (2012) 175–182.
[35] E. Najibi, H. Rostami, Scesn, spesn, swesn: three recurrent neural echo state networks with clustered reservoirs for prediction of nonlinear and chaotic time series, Appl. Intell. 43 (2) (2015) 460–472.
[36] M. Lukoeviius, H. Jaeger, B. Schrauwen, Reservoir computing trends, KI—Knstliche Intell. 26 (4) (2012) 365–371.
[37] D. Koryakin, J. Lohmann, M.V. Butz, Balanced echo state networks, Neural Networks 36 (2012) 35–45.
[38] N. Haghighat, H. Kalbkhani, M.G. Shayesteh, M. Nouri, Variable bit rate video traffic prediction based on kernel least mean square method, IET Image Process. 9 (9) (2015) 777–794.
[39] S. Kang, S. Lee, Y. Won, B. Seong, On-line prediction of nonstationary variable-bit-rate video traffic, IEEE Trans. Signal Process. 58 (3) (2010) 1219–1237.
[40] L. Lu, H. Du, R.P. Liu, Choker: a novel AQM algorithm with proportional bandwidth allocation and TCP protection, IEEE Trans. Ind. Informatics 10 (1) (2014) 637–644.
[41] G.V.D. Auwera, P.T. David, M. Reisslein, Traffic and quality characterization of single-layer video streams encoded with the h.264/mpeg-4 advanced video coding standard and scalable video coding extension, IEEE Trans. Broadcast. 54 (3) (2008) 698–718.
[42] L. Xiang, X.H. Ge, C. Liu, L. Shu, A new hybrid network traffic prediction method, Global Telecommunications Conference, 2010, pp. 1–5.
[43] H. Jaeger, Short term memory in echo state networks, GMD Report, 2002.
[44] Z. Deng, Y. Zhang, Collective behavior of a small-world recurrent neural system with scale-free distribution, IEEE Trans. Neural Networks 18 (5) (2007) 1364–1375.
[45] T. Kuremoto, S. Kimura, K. Kobayashi, M. Obayashi, Time series forecasting using a deep belief network with restricted Boltzmann machines, Neurocomputing 137 (15) (2014) 47–56.
[46] H. Wang, X. Yan, Optimizing the echo state network with a binary particle swarm optimization algorithm, Knowl. Based Syst. 86 (2015) 182–193.

Xiaochuan Sun received the M.S. degree from Guilin University of Electronic Technology, Guilin, China, in 2010, and the Ph.D. degree from Beijing University of Posts and Telecommunications, Beijing, China, 2013, respectively. He is currently a post-doctoral fellow in the School of Information Engineering, Nanjing University of Posts and Telecommunications, Nanjing, China. His current research interests include reservoir computing, neural networks, deep learning architectures, with applications in the domains of time series prediction and nonlinear system identification.



Tao Li received the Ph.D. degree in Computer Science from the Department of Computer Science, University of Rochester, Rochester, NY, in 2004. He is currently a professor in the School of Computer Science at Nanjing University of Posts and Telecommunications. He is also a professor with the School of Computing and Information Sciences, Florida International University, Miami, FL. His research interests include data mining, computing system management, information retrieval, and machine learning. He received the US National Science Foundation (NSF) CAREER Award and multiple IBM Faculty Research Awards.

Yingqi Li received the B.S. and M.S. degrees in Guilin University of Electronic Technology, Guilin, China, in 2007 and 2010, respectively. She has been with the College of Electrical Engineering, North China University of Science and Technology, Tangshan, since 2010, and is currently a lecturer. Her current research interests include pattern recognition, evolutionary computation, neural networks.

Qun Li received the Ph.D. degree in signal and information processing, from Beijing University of Posts and Telecommunications, Beijing, China, in 2013. Now she is with Nanjing University of Posts and Telecommunications, Nanjing, China. Her research interests are in computer vision, pattern recognition and machine learning, with emphasis on image categorization, object recognition and image retrieval.

Yue Huang received the B.S. and M.S. degrees from the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2009 and 2012, respectively. He is currently pursuing the Ph.D. degree with the School of Computer science and Technology, Nanjing University of Posts and Telecommunications, Nanjing, China. His current research interests include neural network models and machine learning methods.

Jiayu Liu received the B.S. and M.S. degrees from the School of Electronic and Information Engineering, Hebei University of Technology, Tianjin, China, in 2003 and 2006, respectively. His current research interests include pattern recognition and intelligent system.