Information Sciences 511 (2020) 152–171

Contents lists available at ScienceDirect

Information Sciences

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

DeePr-ESN: A deep projection-encoding echo-state network

Qianli Ma[a,*], Lifeng Shen[a], Garrison W. Cottrell[b]

[a] School of Computer Science and Engineering, South China University of Technology, Guangzhou, 510006, China
[b] Department of Computer Science and Engineering, University of California, San Diego, CA 92093, USA

ARTICLE INFO

Article history:
Received 16 February 2019
Revised 15 September 2019
Accepted 23 September 2019
Available online 24 September 2019

Keywords:
Hierarchical reservoir computing
Echo state network
Multiscale dynamics
Time series prediction

ABSTRACT

As a recurrent neural network that requires no training, the reservoir computing (RC) model has attracted widespread attention in the last decade, especially in the context of time series prediction. However, most time series have a multiscale structure, which a single-hidden-layer RC model may have difficulty capturing. In this paper, we propose a novel multiple projection-encoding hierarchical reservoir computing framework called Deep Projection-encoding Echo State Network (DeePr-ESN). The most distinctive feature of our model is its ability to learn multiscale dynamics through stacked ESNs, connected via subspace projections. Specifically, when an input time series is projected into the high-dimensional echo-state space of a reservoir, a subsequent encoding layer (e.g., an autoencoder or PCA) projects the echo-state representations into a lower-dimensional feature space. These representations are the principal components of the echo-state representations, which removes the high frequency components of the representations. These can then be processed by another ESN through random connections. By using projection layers and encoding layers alternately, our DeePr-ESN can provide much more robust generalization performance than previous methods, and also fully takes advantage of the temporal kernel property of ESNs to encode the multiscale dynamics of time series. In our experiments, the DeePr-ESNs outperform both standard ESNs and existing hierarchical reservoir computing models on some artificial and real-world time series prediction tasks.

1. Introduction

Reservoir computing (RC) [23,46] is a popular approach for training recurrent neural networks (RNNs) efficiently. They have a “reservoir” of dynamics that can be easily tapped to process complex temporal data. An RC network usually consists of three components: an input layer, a large RNN layer (called the reservoir) and a linear output layer. The input-to-hidden and hidden-to-hidden (recurrent) weights are randomly initialized and fixed during the learning stage. By learning the state dynamics generated from the untrained recurrent hidden layer, reservoir computing networks can avoid the laborious process of gradient-descent RNN training, yet achieve excellent performance in nonlinear system identification [38], signal processing [48] and time series prediction [49]. The computational efficiency presented by reservoir computing networks is highly expected in many practical scenarios, such as some real-time applications [35]. Thus, it attracts many interests in research communities on how to develop new reservoir computing models with high computational efficiency and better modeling capacity.

* Corresponding author.
E-mail address: qianlima@scut.edu.cn (Q. Ma).

https://doi.org/10.1016/j.ins.2019.09.049
0020-0255/

The two most well-known RC models are Liquid State Machines [25] and Echo State Networks [16,19]. Liquid State Machines (LSMs) have continuous dynamics and spiking neurons, while Echo State Networks (ESNs) use discrete dynamics and rate-coded neurons. Due to the relative simplicity and strong theoretical grounding [17,24,28,47,50], ESNs represent the most widely known RC networks and have had many successful applications [22]. The reservoir in an ESN is initialized with sparse connections and a constraint on the spectral radius that guarantees rich, long-term dynamics called the Echo State Property (ESP) [19,23,50]. The reservoir of an ESN can be viewed as a nonlinear temporal kernel, which can map sequences of inputs into a high dimensional space. Learning is then reduced to linear regression from the reservoir to the outputs. Hence, ESNs are a powerful and efficient tool for the representation of dynamic data.

However, on one hand, most of the existing ESNs have limits on modeling the hierarchical multi-scale structure of time series that naturally exist in temporal data [5]. This is mainly because the conventional ESN is a single-hidden-layer RNN, which can have difficulty dealing with input signals that require explicit support for multiple time scales and temporal hierarchy [14]. Therefore, to quote Herbert Jaeger, "*A natural strategy to deal with multiscale input is to design hierarchical information processing systems, where the modules on the various levels in the hierarchy specialize on features on various scales.*" [18].

On the other hand, with the recent revolution in deep learning (DL), several hierarchical RNN architectures have been proposed [5,6,11,14,30]. However, since traditional RNNs depend on the back-propagation through time (BPTT) learning process [33], the lengthy process of training deep RNNs is still a practical issue [30]. Hence, constructing a hierarchical ESN is an attractive approach to this problem as the training is trivial. Nevertheless, constructing a hierarchical ESN-based model while maintaining the stability and high computational efficiency of RC, and also capturing hierarchical multi-scale temporal representation is still a challenging issue.

The first attempt to develop a hierarchical ESN is the dynamical feature discoverer (DFD) model proposed by Jaeger [18]. The main idea is that by stacking multiple reservoirs, the outputs of a higher level in the hierarchy serve as coefficients of mixing (or voting on) outputs from a lower one. The model learns the outputs from each reservoir simultaneously by a gradient-based algorithm, which increases the computational complexity compared to the linear regression training of the original ESN. Subsequently, Triefenbach et al. [43–45] explored cascaded ESNs for speech recognition. The cascaded ESN produces multi-level phonetic states, and can be combined with Hidden Markov Models (HMMs) as the phoneme language model. The cascaded ESN feeds outputs of the previous reservoir into the next one in a supervised way and trains output weights layer by layer.

Most recently, Gallicchio et al. (2016) introduced a stacked echo state network model named deepESN [8,10]. The weights between two reservoirs of a deepESN are randomly generated from a uniform distribution. Doing a visualization experiment similar to one carried out by Hermans and Schrauwen [14], they found empirically that (with some manually tuned hyper-parameters), their deepESN can produce an effective diversification of time-scales over the layers of the hierarchy. Compared with aforementioned DFD and cascaded ESN, the deepESN retains the simplicity of training that the conventional ESN enjoys, and does not need supervised training of each layer. However, these manually designed hyper-parameters are independent of the practical prediction task performance, hence it is unclear whether the multi-scale dynamics in the manually tuned reservoir will result in good performance on time series prediction. In this paper, we will fill this gap and analyze the underlying dynamics of the recent hierarchical ESNs and our model under specific prediction tasks.

Another similar work is the multilayere echo-state machine (MESM) [27]. Unlike the deepESN, the MESM was proposed to pipeline multiple same-size reservoirs. Each reservoir uses the same internal weight matrix, and is connected by subsequent layers by the same coupling weight matrix, which is to ensure the Echo State Property of this hierarchical structure. As with standard ESNs, the output weights of the last reservoir are the only trainable parameters. However, the deepESN and the MESM both simply connect multiple reservoirs with random weights, and can't implement the layer-wise feature learning that is characteristic of deep learning. They are essentially equivalent to conducting multiple random state transitions in the direction of hierarchy, which do not take advantage of hierarchical structure and may not capture rich multiscale dynamics under practical tasks (e.g. time series prediction).

Apart from the above-mentioned ESN-based hierarchical models, some work tries to take advantage of a random-projection technique, the Extreme Learning Machine (ELM) [15], to augment the nonlinearity in reservoirs. One such model is the φ-ESN [7], which is a two-layer model adding a fixed feed-forward layer of ELM to a reservoir. The main idea is to use the ELM to increase the dimensionality over that of the reservoir, in order to increase separability. The other is R^2SP [2,3] which is very similar to the φ-ESN, adding two ELMs to encode inputs and reservoir states respectively. Their results showed that employing some static feed-forward networks as intermediate layers will obtain more nonlinear computational power in the reservoir. Although these works are not typical hierarchical ESN models, they suggest that the suitable allocation of nonlinear mapping capacity and short-term memory capacity on different layers can enhance the modeling capacity of the reservoir system.

In this paper, we propose a novel hierarchical reservoir computing framework called Deep Projection-encoding Echo State Network (DeePr-ESN). A distinctive feature of the DeePr-ESN is its multiple projection-encoding based hierarchical architecture. DeePr-ESN alternates between a projection layer and an encoding layer to connect the reservoirs. More specifically, when an input time series is projected into the echo-state space of a reservoir, a subsequent encoding layer receives the echo states of the previous reservoir as input and encodes the high-dimensional echo-state representations into a lower-dimensional feature space (e.g., by PCA). Then these encoded representations are once again projected into the high-dimensional state space of the following reservoir by random connections. By using this multiple projection-encoding

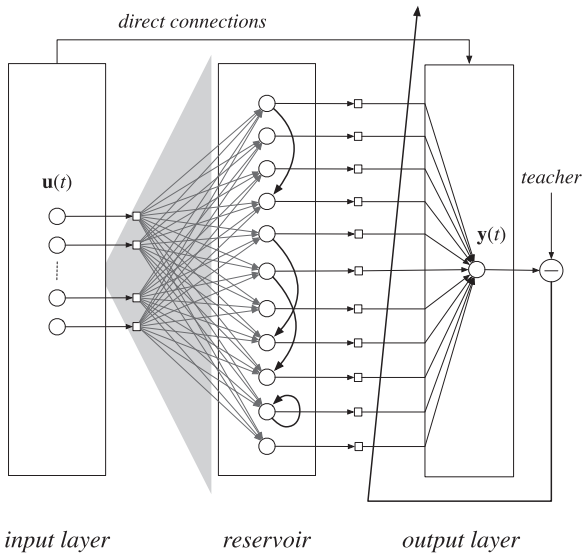


Fig. 1. Architecture of the basic ESN, which consists of three components: an input layer with D neurons, a large reservoir with N neurons, and an output layer with L neurons (here $L = 1$). The links in the reservoir are random sparse fixed connections.

method, the DeePr-ESN can fully take advantage of the temporal kernel property of each reservoir to represent the multiscale dynamics of the time series, rather than directly stacking multiple reservoirs in a entirely random way. Our experimental results show that DeePr-ESN outperforms both the standard ESN and the existing hierarchical ESN models on some artificial and real-world time series prediction by capturing the rich multiscale dynamics of the data.

Our contributions can be summarized as follows:

1. We develop a novel hierarchical reservoir computing framework called the Deep Projection-encoding Echo State Network (DeePr-ESN) based on projection-encodings between reservoirs, which takes advantage of the merits of reservoir computing and deep learning, and bridges the gap between them.
2. By unsupervised encoding of echo states layer by layer, the proposed DeePr-ESN can not only provide more robust generalization performance than existing methods, but also obtains more rich multiscale dynamics than other hierarchical RC models.
3. Compared with the existing Reservoir Computing hierarchical models, the DeePr-ESN achieves better performance on well-known chaotic time series modeling tasks and several real-world time series prediction tasks.

The rest of this paper is organized as follows. In Section II, we introduce the typical ESN architecture, some important properties and related hyper-parameters. In Section III, the model architecture and the learning algorithm of the DeePr-ESN are formally described. After that, we report our experimental results in Section IV. We conclude in Section V.

2. Echo state network

2.1. ESN architecture

An Echo State Network is a recurrent neural network consisting of three basic components: an input layer, a large recurrent layer (called the *reservoir*) with fixed sparse hidden-to-hidden connections, and an output layer [17]. The general architecture of an ESN is illustrated in Fig. 1.

Let D, N and L denote the numbers of neurons in the input layer, the reservoir and the output layer. The weights of input-to-reservoir and reservoir-to-reservoir are collected by an N-by-D matrix $\mathbf{W}^{in}$ and an N-by-N matrix $\mathbf{W}^{res}$. The weights from input-to-output and reservoir-to-output are combined in a single L-by-$(D+N)$ matrix $\mathbf{W}^{out}$. Among these, $\mathbf{W}^{in}$ is randomly initialized from a uniform distribution $[-1, 1]$, and $\mathbf{W}^{res}$ is defined in Eq. (4), below. They are fixed during the training stage, and only $\mathbf{W}^{out}$ needs to be adapted.

An ESN is trained by supervised learning. Two steps are involved. First, a series of D-dimensional inputs $\mathbf{u}$ are projected into the reservoir, driving it to obtain a series of echo states $\mathbf{x}$. Second, the output matrix $\mathbf{W}^{out}$ is learned by simple regression techniques. Here we use an ESN with leaky-integrator neurons as proposed by Jaeger et al. [20], which was also adopted by [8,27]. The mathematical formulation for the entire system is:

$$\mathbf{z}(t+1) = \mathbf{W}^{res}\mathbf{x}(t) + \mathbf{W}^{in}\mathbf{u}(t+1) \tag{1}$$

$$\mathbf{x}(t+1) = (1-\gamma)\mathbf{x}(t) + \gamma f(\mathbf{z}(t+1)) \tag{2}$$

$$\mathbf{y}(t+1) = f^{out}(\mathbf{W}^{out}[\mathbf{x}(t+1); \mathbf{u}(t+1)]) \tag{3}$$

where **u, x** and **y** denote the inputs, the reservoir states and the outputs, respectively. $f(\cdot)$ is the non-linear activation function in reservoir (usually $tanh(\cdot)$) and f^{out} is the activation function in output (usually $identity(\cdot)$). γ in Eq. (2) denotes the leak rate which is used for integrating the states of the previous time step with the current time step.

There are three main characteristics that distinguish an ESN from other RNNs:

1. The ESN adopts a high-dimensional projection process to capture the dynamics of the inputs, which has a similar function to that of the kernel in kernel-based learning methods [13,23];
2. The reservoir is the core of whole system, which consists of a large number (typically 100–1000 times the dimensions of the inputs) of sparsely connected neurons, and none of the weights in the reservoir are trained.
3. The output signals are linear combinations of the echo states of the reservoir (and optionally, the input), and simple linear regression algorithms are used to compute the linear readout layer weights.

Compared with other RNNs, training an ESN is both simple and fast because it need not train all recurrent weights via BPTT and avoids suffering from the issues of gradient exploding and vanishing. Moreover, it cannot get stuck in local minima. The large, sparsely-connected reservoir endows it with a high capacity to model the underlying dynamics of a time series.

2.2. Hyperparameters and initialization

The important hyperparameters used for initializing an ESN are *IS* - the input scaling, *SR* - the spectral radius, α - the sparsity, and the aforementioned leak rate γ.

1. *IS* is used to scale the randomly-generated input matrix $\mathbf{W}^{in}$: the elements of $\mathbf{W}^{in}$ obey a uniform distribution between *-IS* and *IS*.
2. *SR* is the spectral radius of $\mathbf{W}^{res}$, given by
$$\mathbf{W}^{res} = SR \cdot \frac{\mathbf{W}}{\lambda_{max}(\mathbf{W})} \tag{4}$$
where $\lambda_{\max}(\mathbf{W})$ is the largest eigenvalue of matrix **W** and the elements of **W** are generated randomly from $[-0.5, 0.5]$. For practical purposes, *SR* is set below unity to ensure the Echo State Property (ESP) [19,50].
3. α denotes the proportion of non-zero elements in $\mathbf{W}^{res}$. α is set to be 0.1 in this paper.

In short, ESNs have a very simple training procedure, and due to the high-dimensional projection and highly sparse connectivity of neurons in the reservoir, it has abundant non-linear echo states and short-term memory, which are very useful for modeling dynamical systems. The sparsity results in relatively independent nonlinear oscillators of different frequencies. However, a single ESN cannot deal with input signals that require complex hierarchical processing and support for multiple time scales. In the next section, we will propose our hierarchical reservoir-computing framework to resolve this issue.

3. Deep projection-encoding echo state network

In this section, the details of the proposed Deep Projection-encoding Echo State Network (DeePr-ESN) will be described, as well as its learning algorithm, and analyze the stability conditions.

3.1. Proposed hierarchical framework

The architecture of our Deep Projection-encoding Echo State Network (DeePr-ESN) is illustrated in Fig. 2. Its hidden layers consist of K reservoirs and $(K-1)$ encoders. To avoid confusion, we define this DeePr-ESN as a K-layer network (actually it has $2K-1$ hidden layers when the encoders are included). Let $N^{(i)}$ and $M^{(j)}$ denote the number of neurons in the i-th reservoir layer and the j-th encoder layer, where $i = 1, \ldots, K$ and $j = 1, \ldots, K-1$. The T-length time series inputs are denoted as $\mathbf{U} = [\mathbf{u}(1), \mathbf{u}(2), \ldots, \mathbf{u}(T)]$ and teaching signal matrix is denoted as $\mathbf{T} = [\mathbf{t}(1), \mathbf{t}(2), \ldots, \mathbf{t}(T)]$, where $\mathbf{u}(t) \in \mathbb{R}^D$, $\mathbf{t}(t) \in \mathbb{R}^L$ at each time step t. Activation values (updated states) in i-th layer reservoir and j-th layer encoder are denoted as $\mathbf{x}_{res}^{(i)}(t)$ and $\mathbf{x}_{enc}^{(j)}(t)$ respectively, where $t = 0, \ldots, T-1$. Further, We use $\mathbf{W}^{in(i)}$ to collect the input weights of i-th reservoir, $\mathbf{W}^{res(i)}$ to collect the recurrent weights and $\mathbf{W}^{enc(j)}$ to collect the input weights of j-th encoder. Matrix $\mathbf{W}^{out}$ has all the weights of the direct connections, the weights of feature links (connections between intermediate reservoirs and the outputs, as shown in Fig. 2), and the output weights from the last reservoir. In this way, the formulation details of our proposed network can be presented as follows.

For the i-th reservoir ($i = 1, \ldots, K$), its high-dimensional states can be obtained by

$$\mathbf{z}^{(i)}(t+1) = \mathbf{W}^{res(i)}\mathbf{x}_{res}^{(i)}(t) + \mathbf{W}^{in(i)}\mathbf{x}_{in}^{(i)}(t+1) \tag{5}$$

$$\mathbf{x}_{res}^{(i)}(t+1) = (1-\gamma)\mathbf{x}_{res}^{(i)}(t) + \gamma f(\mathbf{z}^{(i)}(t+1)) \tag{6}$$

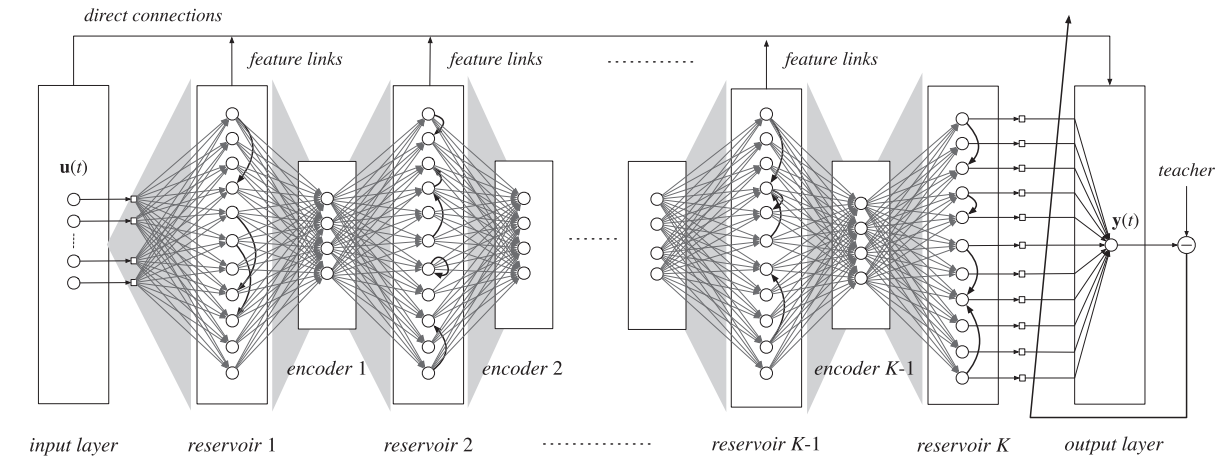


Fig. 2. Architecture of the proposed DeePr-ESN with K reservoirs and $(K-1)$ encoders. We include direct connections from the input to the output layer and weights (feature links) from each reservoir to the output layer.

where $\mathbf{x}_{in}^{(i)}(t+1)$ denotes the inputs of i-th reservoir and $f(\cdot)$ is the non-linear activation function ($tanh(\cdot)$). When i is 1, we have $\mathbf{x}_{in}^{(1)}(t+1) = \mathbf{u}(t+1)$. When i is greater than one, we have $\mathbf{x}_{in}^{(i)}(t+1) = \mathbf{x}_{enc}^{(i-1)}(t+1)$, i.e., the inputs of i-th reservoir are the output of the $(i-1)$-th encoder. For simplicity, we use an operator $\mathcal{F}_i$ to denote the high-dimensional projection (5) and the update step (6). That is

$$\mathbf{x}_{res}^{(i)}(t+1) = \mathcal{F}_i(\mathbf{x}_{res}^{(i)}(t), \mathbf{x}_{in}^{(i)}(t+1)) \tag{7}$$

Given the states of the previous reservoir, we can use an unsupervised dimension reduction technique $\mathcal{T}$ to encode them and produce encoded features. Thus, the encoding procedure of the j-th encoder ($j = 1, \ldots, K-1$) can be formulated as

$$\mathbf{x}_{enc}^{(j)}(t+1) = \mathcal{T}(\mathbf{x}_{res}^{(j)}(t+1)) \tag{8}$$

Further, we can instantiate $\mathcal{T}$ in (8) as

$$\mathcal{T}(\mathbf{x}_{res}^{(j)}(t+1)) = f_{enc}(\mathbf{W}^{enc(j)}\mathbf{x}_{res}^{(j)}(t+1)) \tag{9}$$

where $f_{enc}(\cdot)$ is the activation function of the encoder. When $f_{enc}(\cdot)$ is the identity function, $\mathcal{T}$ is a linear dimensionality reduction technique. The choices of $\mathcal{T}$ will be introduced later.

According to above descriptions, we can obtain the state representations of the last reservoir by

$$\mathbf{x}_{res}^{(K)}(t+1) = \mathcal{F}_K \circ \mathcal{H}_{K-1} \circ \cdots \circ \mathcal{H}_1(\mathbf{u}(t+1)) \tag{10}$$

where $\mathcal{H}_j = \mathcal{T}_j \circ \mathcal{F}_j$ and the symbol $\circ$ denotes a composition operator.

Similar to previous work [8], the DeePr-ESN includes weights from all of the reservoirs to the output layer. We omit weights from the encoder layers to have a comparable number of parameters to previous work. Outputs of the whole system at time $t+1$ can be computed by

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

where $\mathbf{M}(t+1)$ is:

$$\mathbf{M}(t+1) = [\mathbf{u}(t+1)^T, \mathbf{x}_{res}^{(1,\ldots,K)}(t+1)^T]^T \tag{12}$$

which collects the input vector and echo state vectors in $\mathbf{M}$. Rewriting (11) in matrix form, we have:

$$\mathbf{Y} = \mathbf{W}^{out}\mathbf{M} \tag{13}$$

where the columns of $\mathbf{Y}$ and $\mathbf{M}$ range over $1, \ldots, T$.

We use the standard squared error loss:

$$E(\mathbf{W}^{out}) \propto \|\mathbf{T} - \mathbf{Y}\|_2^2 \tag{14}$$

which is a simple regression problem on the parameters $\mathbf{W}^{out}$. $\mathbf{T}$ is a matrix collecting the teacher signals. Since time series present a high-dimensional form (T is larger than the output dimension), this problem always is over-determined and we adopt ridge-regression with Tikhonov regularization [41] to solve it.

$$\hat{\mathbf{W}}^{out} = \mathbf{T}\mathbf{M}^T(\mathbf{M}\mathbf{M}^T + \beta\mathbf{I})^{-1} \tag{15}$$

where β is a small regularization term (such as 10^{-4}).

The whole training procedure of DeePr-ESN are described in Algorithm 1.

Algorithm 1 Updating and Training algorithm of DeePr-ESN.

Input:

U: matrix of input signals; **T**: matrix of teaching signals; T: length of signals; K: number of reservoir layers;D: dim of inputs (default 1); $N^{(i)}$: dim of the i-th reservoir, where $i = 1, \ldots, K$; $M^{(j)}$: dim of the j-th encoder, where $j = 1, \ldots, K-1$; $IS^{(i)}$: the input scaling of the i-th reservoir; $SR^{(i)}$: the spectral radius of the i-th reservoir; $\gamma^{(i)}$: the leak rate of the i-th reservoir; α: the sparsity; β: the regularization term;

Output:

trained output weights $\hat{\mathbf{W}}^{out}$; reservoirs $\mathcal{F}_i$, $i = 1, \ldots, K$; encoders $\mathcal{T}_j$, $j = 1, \ldots, K-1$;

```
generate W_res^(i) and W_in^(i)
for i = 1 to K do
    if i == 1 then
        F_1=generate_reservoir(D, N^(i), IS^(i), SR^(i))
    else
        F_i=generate_reservoir(M^(i-1), N^(i), IS^(i), SR^(i))
    end if
end for
initialize state matrix M;
for i = 1 to K do
    if i < K then
        compute x_res^(i)(t), t = 1, ..., T according to Eq. (7);
        add them to M;
        compute W^enc(i) according to choice of encoder;
        compute x_in^(i+1)(t), t = 1, ..., T;
    else
        compute x_res^(K)(t), t = 1, ..., T according to Eq. (7)
        add them to M;
    end if
end for
compute Ŵ^out = TM^T(MM^T + βI)^-1 Eq. (15)
```

3.2. Choice of encoders

To retain the computational advantages of reservoir computing, the encoder $\mathcal{T}$ should have a low learning cost. We use two simple dimensionality reduction techniques:

(a) *Principal Component Analysis* (PCA) an efficient technique to reduce dimensionality. As a popular DR statistical method, PCA adopts an orthogonal base transformation to project the observations into a linearly uncorrelated low-dimensional representation where the selected orthogonal bases are called *principal components*. In mathematical terms, PCA attempts to find a linear mapping $\mathbf{W} \in \mathbb{R}^{D\times M}$ $(M < D)$ that maximizes the following optimization problem:

$$\mathbf{W}^{\star} = \arg\max_{\mathbf{W}} \|\mathbf{W}^T \mathbf{S}_x \mathbf{W}\|_2 \tag{16}$$

$$subject\ to\ \ \mathbf{W}^T\mathbf{W} = \mathbf{I}_M \tag{17}$$

where $\mathbf{S}_x = \mathbf{X}\mathbf{X}^T$ is the covariance matrix of zero-mean inputs $\mathbf{X} \in \mathbb{R}^{D\times N}$. $\mathbf{I}_M$ is the $M \times M$ identity matrix. D and M are the original and reduced dimension of $\mathbf{X}$ respectively. The optimal $\mathbf{W}^{\star}$ can be provided by eigenvectors corresponding to the m largest eigenvalues of the covariance matrix $\mathbf{S}_x$.
While standard PCA is dominated by the eigenvalue decomposition, and so is $\mathcal{O}(N^2T + N^3)$ [26], where N is the original dimensionality and T is the number of data points, there are fast, iterative methods that are $\mathcal{O}(N^2T + N^2MI)$, where M is the reduced dimension, and I is the number of iterations to converge, which is usually quite small (usually 2–5), and therefore the complexity can be rewritten to $\mathcal{O}(N^2T + N^2M)$ [34].

(b) *ELM-based Auto-encoder* (ELM-AE) [4,40] is a recent dimensionality reduction tool based on the Extreme Learning Machine (ELM), which is used for simplifying the training of traditional auto-encoders. The main idea is to obtain the hidden random features $\mathbf{H} \in \mathbb{R}^{M\times T}$ by using random weights $\mathbf{W}^0 \in \mathbb{R}^{M\times N}$ and random bias $\mathbf{b}^0 \in \mathbb{R}^{M\times T}$, formulated by

$$\mathbf{H} = g(\mathbf{W}^0\mathbf{X} + \mathbf{b}^0) \tag{18}$$

where $\mathbf{X} \in \mathbb{R}^{N\times T}$ is the inputs and g denotes the identity activation function. Then the dimension reduction mappings $\mathbf{W}^{\star} \in \mathbb{R}^{N\times M}$ can be solved by optimizing the following problem:

$$\mathbf{W}^{\star} = \arg\min_{\mathbf{W}} \{\|\mathbf{W}\mathbf{H} - \mathbf{X}\|_2 + \lambda\|\mathbf{W}\|_2\} \tag{19}$$

where λ is the regularization coefficient. This problem is a simple regression problem and can be solved by the pseudo-inverse technique, same as (15). Finally, the reduced data $\mathbf{H}_{enc}$ can be represented by $\mathbf{H}_{enc} = (\mathbf{W}^{\star})^T\mathbf{X}$. The computational complexity of (19) is $\mathcal{O}(NMT)$.

3.3. Heuristic optimization of hyperparameters

As the depth of hierarchical reservoir computing models increases, it becomes more important to optimize the hyperparameters, since there are more reservoirs in the models. Existing hierarchical ESN-based research seldom discusses this issue. Here we consider several methods for this element of the modeling enterprise.

There are three commonly used strategies for optimizing the hyperparameters: manual setting (based on the user's experience), grid search, and heuristic optimization. The first two strategies are used for general single-reservoir ESNs, but they are unsuitable for deep reservoir networks due to the increasing dimensionality of the parameter space. Thus, we use heuristic optimization to learn these key hyperparameters. For this purpose, we use the genetic algorithm (GA), as it has been shown to work well in high-dimensional spaces [29].

The GA works on a population of candidate solutions. First, the fitness of every individual in the population is evaluated in each iteration (called a "generation"), where the fitness is the value of the objective function. The more fit individuals are stochastically selected from the current population and used to form a new generation by three biologically-inspired operators: mutation, crossover and selection. Finally, the algorithm terminates when a maximum number of generations or a satisfactory fitness level is reached.

In our experiments, we use the GA optimize three hyperparameters in each reservoir: *IS, SR* and γ. We concatenate all triple hyperparameters of each reservoir into a vector and view this vector as an individual in GA. We restrict the search space for *IS* to the range [0.00001,1], where we use a small value 0.00001 to avoid *IS* becoming zero. *SR* and γ are limited to (0,1). *SR* is set to be smaller than 1 to ensure the Echo State Property in each reservoir (see [9] for more information about the Echo State Property). In the case of our DeePr-ESN, we can derive a similar property, with the restriction that the norms of the intermediate encoder's weights should be bounded. The hyper-parameter α (the sparsity), gives relatively stable results over a wide range in our experiments, as we show later. Hence we fix it to be 0.1 in each reservoir. The prediction error of the system is selected as the fitness value of individuals (the smaller the loss, the higher the fitness). In the following experiments, we apply the GA API provided by the MATLAB Global Optimization Toolbox. Other settings are set to their defaults (e.g., migration fraction, crossover fraction). All of the hyper-parameters are found by GA by evaluation on the validation set, and are listed in our supplementary material.

Remark 1. Here we highlight the main differences between our model and its closest relatives in the literature, deepESN [8] and MSEM [27].

(1) In our work, we mainly focus on developing an efficient mechanism for hierarchical reservoir computing models. This requires propagating useful information to subsequent reservoirs. However, deepESN and MESM both simply connect multiple reservoirs with random weights. Our model uses simple features learned in an unsupervised way (e.g., PCA). We think of this as our lightweight alternative to the feature learning in deep networks. Hence, our model retains the minimal computational requirements of ESNs due to the linearity and simplicity of the encoders.
(2) The authors of deepESN analyze the effectiveness of their network's hierarchical processing by observing multiscale dynamics with specific hyper-parameters, which are manually set and independent of the task. In contrast, our model's rich multiscale dynamics produces high prediction performance. We also find that the deepESN doesn't achieve these multiscale characteristics with optimized hyper-parameters.
(3) The authors of deepESN and MESM, do not address the issue of optimizing the hyperparameters of their models. The number of hyperparameters in their models increases with depth. To address this problem, we use the genetic algorithm (GA) to optimize some of the important hyperparameters of reservoirs. This allows our hierarchical ESN model to adapt to different tasks without manual intervention.

4. Experiments

In this section, we evaluate our model against several single-layer ESNs, and several previous hierarchical ESN models, on four time series datasets. Then we evaluate the model's sensitivity to hyperparameters, and finally, visualize the multiscale dynamics of the DeePr-ESN.

4.1. Experimental setup

4.1.1. Time series data

We used four different time series to evaluate our model, two synthetic and two real-world datasets, shown in Fig. 3.

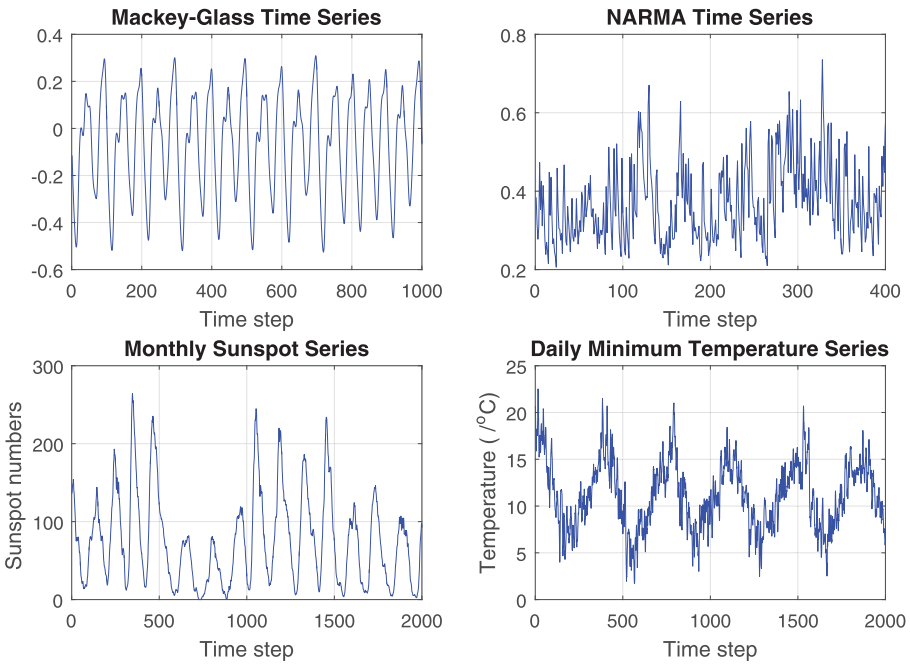


Fig. 3. Visualization of selected time series, including chaotic time series generated from MGS and NARMA system, real world time series (monthly sunspot series and daily minimum temperatures).

Mackey-Glass System (MGS). MGS is a classical chaotic system that is often used evaluating time series prediction models [16]. We use a standard discrete time approximation to the Mackey-Glass delay differential equation given by

$$y(t+1) = y(t) + \delta \cdot (a\frac{y(t-\tau/\delta)}{1+y(t-\tau/\delta)^n} - by(t)) \tag{20}$$

where the parameters δ, a, b, n are usually set to 0.1, 0.2, -0.1, 10. When $\tau > 16.8$, the system becomes chaotic, and in most previous work, τ is set to 17, which we use here. The initial $y(0)$ is set to be 1.2. We generate three parts of MG time series with length $T_{\text{train}} = 10000$, $T_{\text{validate}} = 300$ and $T_{\text{test}} = 300$. To avoid the influence of initial states, we use a burn-in period of 100 steps ($T_{\text{burn-in}} = 100$ for each reservoir). The task is to do 84-step-ahead direct prediction. Thus, all of the methods are required to learn the mapping from the current input $\mathbf{u}(t)$ to the target output $\mathbf{y}(t)$, where $\mathbf{y}(t) = \mathbf{u}(t+84)$.

Nonlinear auto-regressive moving average system (NARMA). NARMA is a highly nonlinear system incorporating memory. The tenth-order NARMA system identification problem is often used to test ESN models. A tenth-order NARMA system depends on outputs and inputs 9 time steps back, so it is considered difficult to identify. The tenth-order NARMA system that we use is:

$$y(t+1) = 0.3 \cdot y(t) + 0.05 \cdot y(t) \cdot \sum_{i=0}^{9} y(t-i) + 1.5 \cdot u(t-9) \cdot u(t) + 0.1 \tag{21}$$

where $u(t)$ is a random input of the system at time step t, drawn from a uniform distribution over [0, 0.5]. The output signal $y(t)$ is initialized by zeros for the first ten steps ($t = 1, 2, \ldots, 10$). In this task, we generate a NARMA time series with three parts with length $T_{\text{train}} = 2000$, $T_{\text{validate}} = 500$ and $T_{\text{test}} = 500$, respectively. The initial burn-in length is set to be 30 for each reservoir in the models. We conduct one-step-ahead direct prediction on this time series.

Monthly Sunspot Series Forecasting. Sunspots are a dynamic manifestation of the strong magnetic field in the Sun's outer regions. It has been found that the number of sunspots has a very close statistical relationship with solar activity [1]. Due to the complexity of the underlying solar activity and the high nonlinearity of the sunspot series, forecasting and analyzing these series is a challenging task commonly used to evaluate the capability of time-series prediction models. An open source 13-month smoothed monthly sunspot series is provided by the world data center SILSO [37]. There are a total of 3209 sample points from July, 1749 to November, 2016. Since the last 11 points are still provisional and subject to possible revision, we remove these points and use the remaining 3198 observations. We select data with three parts: $T_{\text{train}} = 800$ (March, 1756 to October, 1822), $T_{\text{validate}} = 300$ (July, 1829 to June, 1854) and $T_{\text{test}} = 300$ (March, 1861 to February, 1886), respectively. The additional burn-in length T_{burnin} for each reservoir is set to 10. For this data set, we rescale the observations by dividing them by 1000. Here, we also perform one-step-ahead prediction.

Daily Minimum Temperatures in Melbourne Australia. This data is recorded from January 1, 1981 to December 31, 1990 [42]. There are a total of 3650 sample points. We select the data with three parts: $T_{\text{train}} = 800$ (March 22, 1981 to May

30, 1983), $T_{\text{validate}} = 300$ (August 19, 1983 to June 13, 1984) and $T_{\text{test}} = 300$ (September 02, 1984 to June 29, 1985). The additional burn-in length T_{burnin} for each reservoir is set to 10. Since this real world time series shows strong nonlinearity, we smooth it with a 5-step sliding window. The smoothed data can be seen in Fig. 3. We perform one-step-ahead direct prediction on this data set.

4.1.2. Comparison models

We compare our DeePr-ESN with the standard single-reservoir ESN and several hierarchical models. There are many variants of single-reservoir ESNs, such as the Simple Cycle Reservoir (SCR) [32], the support vector echo state machine (SVESM) [36], and the recent incremental ESN variant called the Growing ESN (GESN) [31]. Since we focus on exploring and analyzing the effectiveness of hierarchical schema, we ignore these single-reservoir variants, but we note here that they can all be used as a module in our hierarchical framework.

The hierarchical models we use are two related two-layer models, including φ-ESN [7], R^{2}SP [2,3], and the recent hierarchical ESN-based models, including the multilayered echo-state machine (MESM) [27], the deepESN and grouped-ESN proposed by Gallicchio et al. [8,10]. The grouped-ESN model actually is not a hierarchical model, but simply uses multiple independent reservoirs, and the linear regression is performed on all the reservoirs simultaneously.

Note that in the original papers, these researchers did not use the GA to choose their hyperparameters. In order to ensure fairness of comparison, in this paper the GA is employed to optimize the hyperparameters in all the models. We set the population size to 20 individuals and apply the GA for 60 generations.

4.1.3. Comparison fairness

For doing further fair comparison, we also employ identical conditions on the number of trained parameters in the output layer of all hierarchical ESNs. First, we fix the size of each reservoir of all models to be 300, which is a reasonable size relative to the dimension of the input variable, which is univariate. Then, we compare all hierarchical models with increasing layers. Note that in our experiments, we add connections between intermediate reservoirs and the outputs in all models. This ensures that the output weight matrix $\mathbf{W}^{out}$ in the regression problem (15) is identical for all hierarchical ESNs.

For the standard ESN, we set various reservoir sizes to keep the same aggregate number of units in all reservoirs in the hierarchical models.

We do not include any connections from inputs to the intermediate reservoirs (as the case of deepESN-In All), because we want to analyze merely the performance of different pipeline-style hierarchical structures. This would benefit the observation of the importance of various layers. Including connections from inputs to the intermediate would probably make hierarchical model degenerate to a single-layer ESN.

4.1.4. Evaluation metric

The performance of all methods is evaluated by the normalized root mean squared error (NRMSE), which is used by most ESN-based methods and is formulated as follows:

$$\text{NRMSE} = \sqrt{\sum_{t=1}^{T}[\mathbf{d}(t) - \mathbf{y}(t)]^2 / \left\{\sum_{t=1}^{T}[\mathbf{d}(t) - \bar{\mathbf{d}}]^2\right\}} \tag{22}$$

where $\mathbf{d}(t)$ denotes the teaching signal at time step t, $\mathbf{y}(t)$ denotes t-th output of the model, and $\bar{\mathbf{d}}$ denotes the mean of the teaching signals, and T is the number of points. A nice property of this measure is that it is 1 if the system just produces the average value of the teaching signal, and achieves its minimum at 0 if the fit is perfect. It is worth emphasizing that there are two prediction methodologies, *iterated prediction* and *direct prediction*. The former estimates the next value by the previous predicted one, while the latter finds a mapping between the input $\mathbf{u}(t)$ and the output $\mathbf{u}(t+h)$ in the h-step-head direct prediction. Although the iterated prediction was employed in the original ESN [16] and had higher accuracy than the direct one [36], previous work [12,36,39,49] found iterated prediction will lead to instability due to the error accumulation for real-world nonlinear time series. Therefore, we employ the *direct prediction* in the following experiments.

All results reported here are averages over 20 runs with different random reservoir initialization.

4.2. Results

The results are shown in Tables 1–4. We use a color code (green is worst, red is best) so that the best results are easily seen, and the trends as the number of layers increases can be readily observed. Note that in all tables, as the number of layers increases, for the single-reservoir ESN, only the number of parameters increases. For the two-layer ESNs, φ-ESN [7] and R^{2}SP [2,3], the size of the two reservoirs are increased to provide a matched number of parameters. For the grouped-ESN, which is simply a flat combination of multiple reservoirs, it is the number of reservoirs that increases from left to right. For DeePr-ESN, the encoder type and the encoder dimension, optimized by grid search, is shown in parentheses.

The results on the Mackey-Glass dataset are reported in Table 1. The Mackey-Glass results for DeePr-ESN are better than all other models, with the ELMAE variant performing the best. The next best result overall is for grouped-ESN with 6 reservoirs

Table 1
Performance on Mackey-Glass time series.

\# of layers		2	3	4	5	6	7	8
\# of trained parms		600	900	1200	1500	1800	2100	2400
ESN	Mean.	8.29E-02	1.12E-01	5.31E-02	4.79E-02	5.87E-02	7.49E-02	2.30E-01
	Std.	2.54E-02	4.86E-02	2.75E-02	1.11E-02	2.17E-02	3.55E-02	9.25E-02
φ-ESN	Mean.	9.40E-02	8.54E-02	1.19E-01	1.96E-01	1.47E-01	2.31E-01	1.17E-01
	Std.	1.99E-02	1.45E-02	5.58E-03	8.89E-03	2.53E-03	9.39E-03	2.09E-02
R^2SP	Mean.	5.46E-02	3.55E-02	2.65E-02	2.87E-02	2.55E-02	2.22E-02	2.14E-02
	Std.	1.29E-02	4.13E-03	3.39E-03	2.94E-03	1.83E-03	1.85E-03	2.03E-03
grouped-ESN	Mean.	3.62E-02	2.37E-02	1.47E-02	1.18E-02	9.73E-03	3.75E-02	3.20E-02
	Std.	2.90E-03	2.89E-03	3.66E-03	2.67E-03	2.81E-03	5.30E-03	6.58E-03
MESM	Mean.	3.91E-02	2.43E-02	1.68E-02	2.15E-02	1.55E-02	1.72E-02	1.94E-02
	Std.	6.83E-03	4.30E-03	2.25E-03	3.08E-03	2.44E-03	1.36E-03	1.79E-03
deepESN	Mean.	2.63E-02	2.14E-02	2.72E-02	2.21E-02	1.71E-02	2.32E-02	2.37E-02
	Std.	3.11E-03	2.52E-03	4.16E-03	1.66E-03	1.98E-03	1.58E-03	1.40E-03
DeePr-ESN (PCA, 60)	Mean.	2.79E-02	1.62E-02	1.23E-02	9.41E-03	8.59E-03	8.66E-03	8.14E-03
	Std.	2.38E-03	3.04E-03	1.57E-03	2.59E-03	4.02E-03	3.88E-03	2.18E-03
DeePr-ESN (ELMAE, 60)	Mean.	2.74E-02	1.66E-02	5.00E-03	4.31E-03	1.71E-03	1.63E-03	9.08E-04
	Std.	4.65E-03	8.76E-03	1.52E-03	2.80E-03	8.37E-04	8.95E-04	9.67E-04

Table 2
Performance on NARMA time series.

\# of layers		2	3	4	5	6	7	8
\# of trained parms		600	900	1200	1500	1800	2100	2400
ESN	Mean.	1.97E-01	1.44E-01	1.36E-01	1.24E-01	1.25E-01	1.25E-01	1.24E-01
	Std.	1.20E-02	1.33E-02	1.12E-02	1.34E-02	1.19E-02	7.74E-03	5.73E-03
φ-ESN	Mean.	1.78E-01	1.86E-01	1.34E-01	1.38E-01	1.31E-01	1.39E-01	1.45E-01
	Std.	1.39E-02	1.35E-02	1.78E-02	1.38E-02	1.58E-02	1.98E-02	1.69E-02
R^2SP	Mean.	1.98E-01	1.80E-01	1.55E-01	1.31E-01	1.50E-01	1.53E-01	1.37E-01
	Std.	2.37E-02	2.87E-02	1.83E-02	1.33E-02	1.25E-02	1.32E-02	7.66E-03
grouped-ESN	Mean.	1.98E-01	1.39E-01	1.45E-01	1.38E-01	1.32E-01	1.34E-01	1.27E-01
	Std.	2.77E-02	1.39E-02	1.40E-02	1.52E-02	9.71E-03	1.27E-02	1.02E-02
MESM	Mean.	1.59E-01	1.48E-01	1.40E-01	1.23E-01	1.17E-01	1.21E-01	1.31E-01
	Std.	3.70E-02	3.13E-02	2.51E-02	1.59E-02	1.96E-02	2.04E-02	2.64E-02
deepESN	Mean.	1.25E-01	1.24E-01	1.21E-01	1.12E-01	1.15E-01	1.05E-01	1.20E-01
	Std.	2.29E-02	6.85E-02	1.53E-02	1.03E-02	1.56E-02	1.23E-02	1.60E-02
DeePr-ESN (PCA, 300)	Mean.	1.21E-01	1.23E-01	1.19E-01	1.16E-01	1.14E-01	1.02E-01	1.15E-01
	Std.	1.33E-02	1.50E-02	1.41E-02	9.63E-03	1.42E-02	1.23E-02	1.44E-02
DeePr-ESN (ELMAE, 200)	Mean.	1.05E-01	9.07E-02	9.89E-02	9.46E-02	9.02E-02	6.93E-02	8.20E-02
	Std.	2.09E-02	1.49E-02	8.61E-03	1.02E-02	9.32E-03	1.15E-02	1.26E-02

We found that as the number of layers increases, the DeePr-ESN consistently improves. With 8 reservoirs, DeePr-ESN with PCA and ELMAE achieve their best performance, with NRMSE of 8.14E-03 and 9.08E-04, respectively. Also rather satisfying is that the DeePr-ESN with ELMAE displays the lowest variance in performance at the deepest setting. These results suggest that our DeePr-ESNs can take full advantage of the alternation between projection to low dimension and expansion to higher-dimensional reservoirs to model the time series at multiple scales.

Unfortunately, MESM and deepESN do not enjoy such stable behavior; both achieve their best results with 6 layers and perform comparably to each other. The standard ESN has a sweet spot at a reservoir size of 1500, with worse performance for smaller and larger reservoirs. Similarly, the two layer and grouped-ESN show variable behavior as the number of parameters is increased.

We also observe that DeePr-ESN with ELMAE consistently outperforms DeePr-ESN with PCA. It is partially because PCA expects the input distribution to approximate multivariate Gaussian distribution, while ELMAE seems to have fewer restrictions on the input distribution since it only requires that its inputs can be reconstructed well by outputs. Moreover, from our empirical results, ELMAE is more suitable to be used as the DR layer in our DeePr-ESN. Since ELMAE is more efficient than PCA, this is an especially nice result. In fact, on all of the datasets, ELMAE outperforms PCA for nearly all settings. Although the difference is small for the real-world datasets, the efficiency of ELMAE suggests it is the best choice of encoder.

Table 2 presents the results on NARMA time series. Here, the DeePr-ESN with ELMAE outperforms the other models, while DeePr-ESN with PCA and DeepESN perform similarly across the board. Here, although the improvement of DeePr-ESN (with ELMAE) as the number of layers increases is somewhat less consistent, the general trend is deeper is better, with 7 layers being optimal for this data.

Tables 3 and 4 show the results on the two real-world datasets. Here, the Here, the trends are rather different than for the first two datasets: Two layers are optimal. The fact that this is consistent across all four hierarchical models suggests that it is a characteristic of these datasets that two scales suffice. Also, we note here that while numerically, both DeePr-ESN

Table 3
Performance on sunspot time series.

\# of layers		2	3	4
\# of trained parms		600	900	1200
ESN	Mean.	5.05E-02	4.67E-02	3.18E-02
	Std.	3.69E-03	7.44E-04	9.00E-04
φ-ESN	Mean.	3.76E-02	2.42E-02	3.01E-02
	Std.	3.47E-03	1.41E-03	1.60E-03
R^2SP	Mean.	6.09E-02	6.85E-02	5.19E-02
	Std.	4.85E-03	3.17E-03	2.32E-03
grouped-ESN	Mean.	5.23E-02	5.95E-02	4.78E-02
	Std.	2.82E-04	3.39E-04	4.03E-04
MESM	Mean.	1.95E-02	1.99E-02	2.05E-02
	Std.	5.69E-04	3.47E-04	8.30E-02
deepESN	Mean.	2.02E-02	2.15E-02	4.06E-02
	Std.	4.42E-03	5.71E-03	3.14E-03
DeePr-ESN (PCA, 120)	Mean.	1.90E-02	2.03E-02	2.24E-02
	Std.	3.02E-04	7.81E-04	8.41E-04
DeePr-ESN (ELMAE, 120)	Mean.	1.89E-02	2.03E-02	2.07E-02
	Std.	4.49E-04	6.93E-04	3.78E-04

Table 4
Performance on temperature time series.

# of layers		2	3	4
# of trained parms		600	900	1200
ESN	Mean.	1.60E-01	1.86E-01	1.53E-01
	Std.	1.01E-03	1.75E-03	8.30E-04
φ-ESN	Mean.	1.59E-01	1.51E-01	1.64E-01
	Std.	4.17E-03	1.08E-03	1.14E-03
R^2SP	Mean.	1.67E-01	1.56E-01	1.82E-01
	Std.	1.34E-03	1.44E-03	2.38E-03
grouped-ESN	Mean.	1.63E-01	1.56E-01	1.69E-01
	Std.	2.42E-03	1.28E-03	1.58E-03
MESM	Mean.	1.28E-01	1.40E-01	1.45E-01
	Std.	1.11E-03	1.36E-03	1.15E-03
deepESN	Mean.	1.28E-01	1.33E-01	1.48E-01
	Std.	3.63E-03	1.06E-03	4.35E-03
DeePr-ESN (PCA, 90)	Mean.	1.26E-01	1.31E-01	1.49E-01
	Std.	1.55E-03	2.32E-03	3.60E-03
DeePr-ESN (ELMAE, 200)	Mean.	1.27E-01	1.31E-01	1.43E-01
	Std.	1.89E-03	1.26E-03	1.07E-03

models achieve the best results, the differences in performance between the four hierarchical models are small. Overall, these results suggest that synthetic time series, without real-world noise, may not be the best arbiter of performance for time series models in practical applications.

4.3. Effects of hyper-parameters

The four hyper-parameters, the input scaling *IS*, the spectral radius *SR*, the leak rate γ and the sparsity α all influence the performance of the reservoirs in DeePr-ESN. In addition, the regularization term β used in the regression will impact the learning of the output weights. Since investigating how all four of these interact is impractical, in this section, we investigate the results of varying each of them independently while holding the others fixed at their optimal values.

We investigated the effects of the hyperparameters using a two-layer DeePr-ESN with one ELMAE encoder between them, while varying the regularization term β over four orders of magnitude. We plot the NRMSE as a function of the parameter of interest in the two reservoirs. The results are shown in Fig. 4. The β parameter is decreased over each column from left to right, and each row corresponds to a different hyperparameter. The right x-axis corresponds to the parameter for the first reservoir, and the left x-axis for the parameter for the second reservoir. The parameters are maximum at the front corner and get smaller along each axis.

The first thing to notice is that when the regularization parameter is too small (right column), performance degrades catastrophically when the other parameters are small. There is an especially strong interaction between the sparsity parameter (α) for the first reservoir and β, even at the relatively common setting of $\alpha = 0.1$. This setting of α leads to more independent oscillators in the reservoir, and suggests that these may overfit the training data without the weights from them being regularized. On the other hand, there is a wide range of α values (roughly 0.3 to 1.0) where performance is stable. It's remarkable that this extends all the way to $\alpha = 1$, as that means that the reservoir works well fully connected.

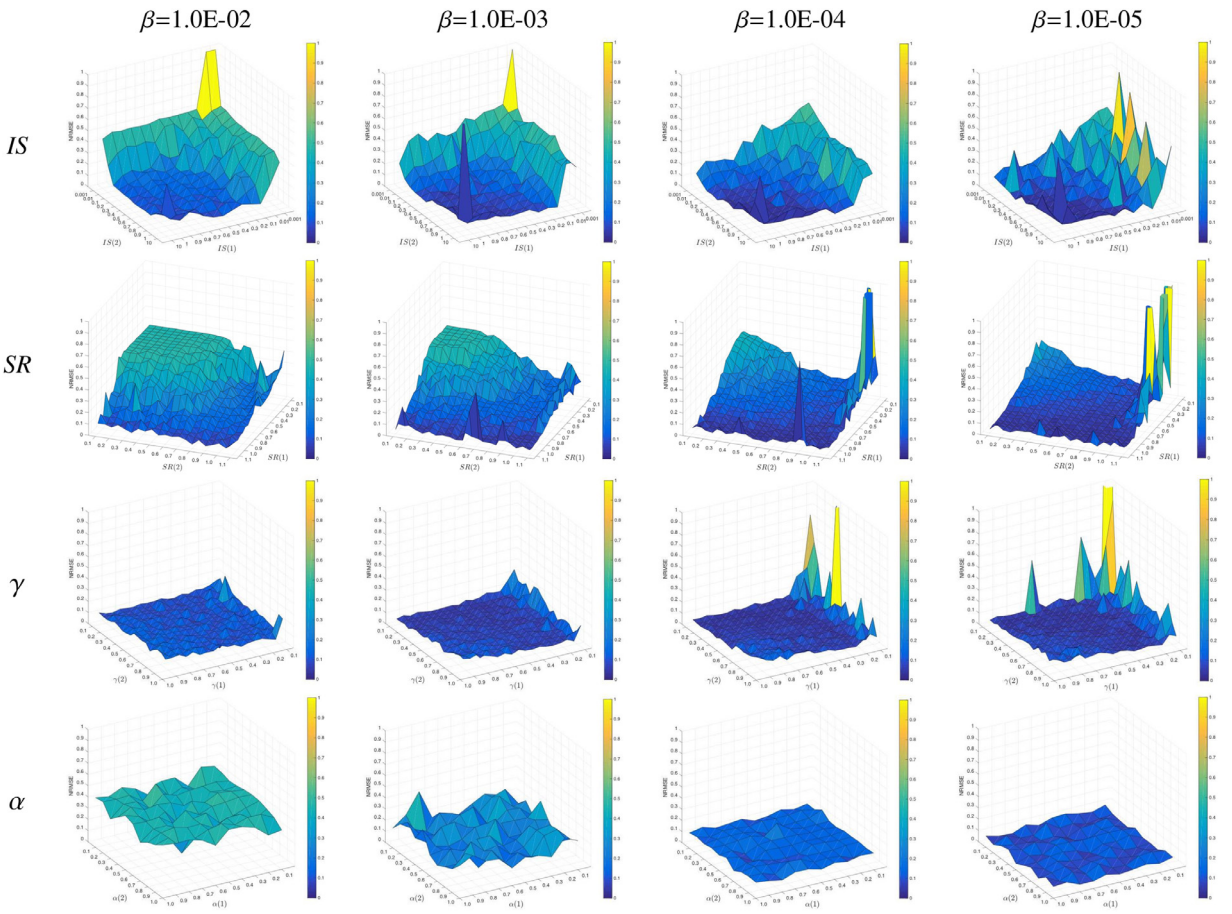


Fig. 4. Visualization of error surfaces (lower is better) under various hyperparameter settings (with different regularization term β).

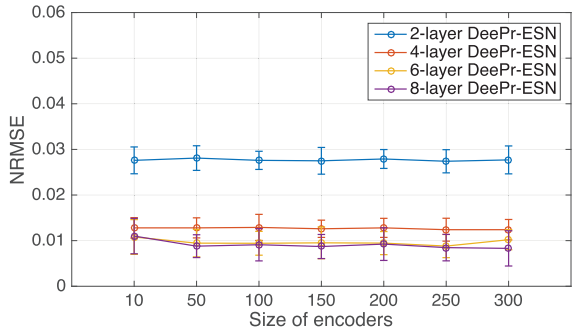
All the other hyper-parameters are fixed with the optimized values obtained from the MGS-84 prediction task. We found that the performance of our DeePr-ESN tends to be better and more stable with the *IS* and *SR* in the range of 0.5 to 1. When they are much smaller or larger than unit, the performance is worse. For the hyper-parameter of the input scaling, *IS*(1) is more important than *IS*(2) since *IS*(1) impacts on input signals directly.

Compared with the *IS* and *SR*, the performance of DeePr-ESN is relatively stable to variations in the leak rate γ and sparsity α (see the 3rd-4th rows in Fig. 4). When the γ is smaller than 0.3 or larger than 0.7, the performance on this task tends to be worse. α performs much more stable when the regularization term β is correctly selected (e.g., 1.0E-04 or 1.0E-05).

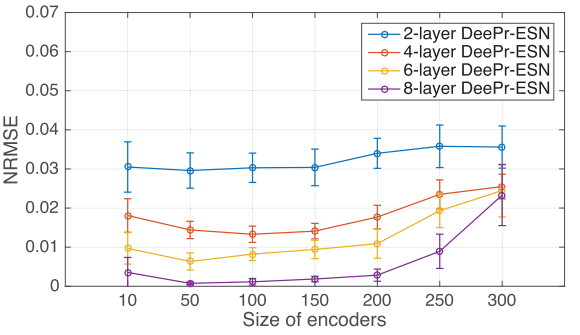
Apart from above four hyper-parameters, the size of reservoir and encoder also play important roles in designing a hierarchical reservoir-computing system. Therefore, we also investigated them here.

First, we varied the size of the encoders while holding the reservoir size fixed at 300 (hence the setting of 300 corresponds to no information loss). Fig. 5 shows the performance of DeePr-ESNs with different number of layers when we vary the size of encoders on the 84-step-ahead Mackey-Glass prediction task and the 1-step-ahead NARMA prediction task. We found that on the Mackey-Glass task, there is little effect of the encoder size for PCA, suggesting the main role of PCA is to decorrelate the variables. ELMAE is more sensitive to encoder size, and performs worst when there is no dimensionality reduction, while the depth has very little effect. The lack of a depth-of-encoder effect suggests that the NARMA task has little in the way of multi-scale dynamics. As long as enough information gets through (i.e., the encoder dimension is 50 or greater), the ELMAE encoder does well.

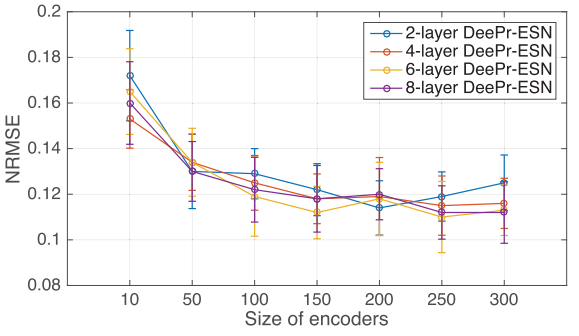
We also investigated the effects of the size of the reservoirs in DeePr-ESNs. On the Mackey-Glass task, we fix the size of encoders to 60, as was used in Table 1. On the NARMA task we use the encoder size of 100. As seen in Fig. 6, as long as the reservoir is large enough, DeePr-ESNs will tend to perform better and are stable. Consistent with Fig. 5, depth does not appear to matter for the NARMA task, again, as long as the reservoir is big enough. It is difficult to see much difference in the graph, but for Mackey-Glass, when the size of reservoir reaches 1200, the performance of the 6-layer DeePr-ESN with ELMAE achieves and NRMSE of 8.21E-05, which is an order of magnitude better than our best result in Table 1 (NRMSE=9.08E-04).



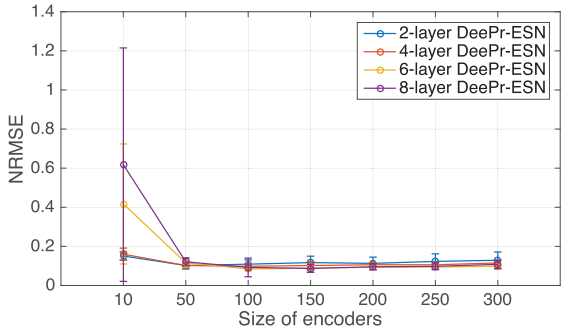
(1) DeePr-ESN with PCA on MG task



(2) DeePr-ESN with ELMAE on MG task

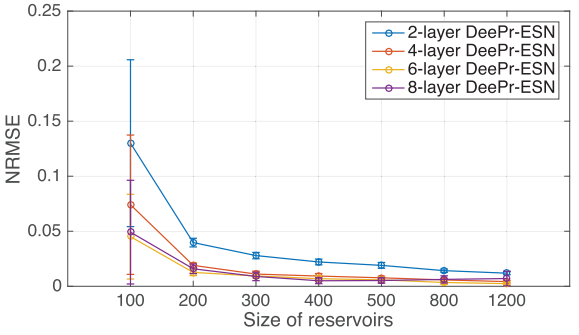


(3) DeePr-ESN with PCA on NARMA task

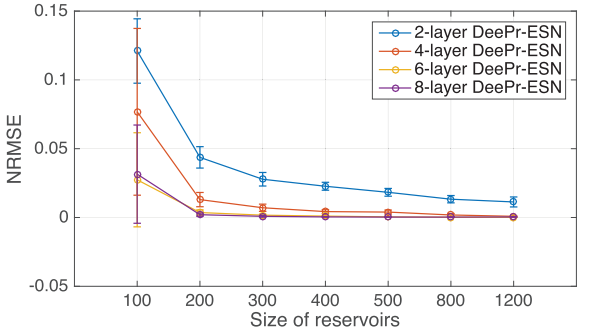


(4) DeePr-ESN with ELMAE on NARMA task

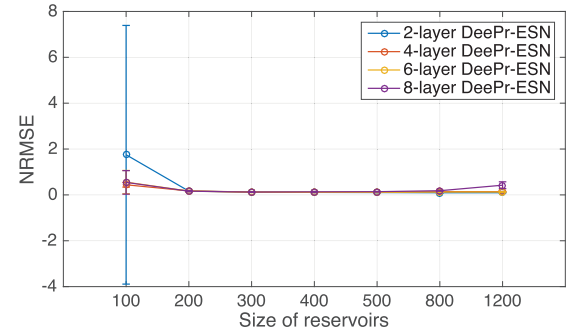
Fig. 5. Investigation on the effect of varying size of encoder in DeePr-ESNs.



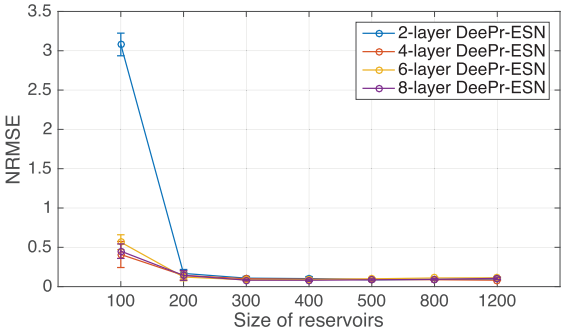
(1) DeePr-ESN with PCA on MG task



(2) DeePr-ESN with ELMAE on MG task



(3) DeePr-ESN with PCA on NARMA task



(4) DeePr-ESN with ELMAE on NARMA task

Fig. 6. Investigation on the effect of varying size of reservoir in DeePr-ESNs.

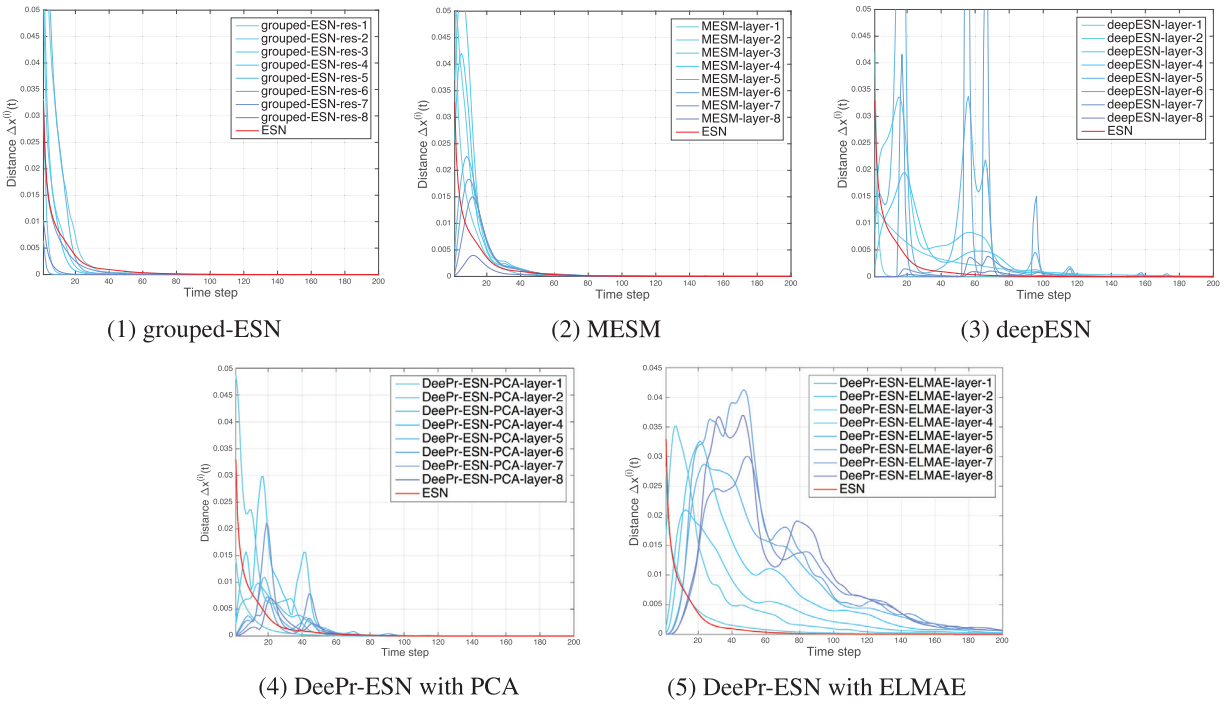


Fig. 7. Visualization of the multiscale dynamics on Mackey-Glass 84-step ahead direct prediction task. The staring point at x-axis is corresponding to the perturbed time 200.

4.4. Visualization of multiscale dynamics

Visualizing multiscale dynamics is one of the most intuitive and effective ways to understand the internal mechanism of a given hierarchical system. In this section, we perform an experiment similar to the work of [14] to evaluate the multiscale characterises within a hierarchical system, where we perturb the system and plot the effect of the perturbation compared to an unperturbed system. We perform this experiment with DeePr-ESN, grouped-ESN, deepESN and MESM. All models are optimized with 8 layers (or 8 reservoirs). This method is also adopted in [8,10]. In the work of [8], the authors analyze the multiscale dynamics by setting different hyperparameters. However, their analysis is independent of practical prediction tasks and they did not present the multi-scale dynamics on time series modeling tasks. We fill this gap in this section and visualize the multiscale dynamics with the optimized hyper-parameter configuration obtained from our experiments.

The data we use here are the Mackey-Glass (MG in the figures) and NARMA time series. We first generate a time series S with the length of 1000. We then perturb S by adding a noise value to the time series at time step 200, which is denoted as S'. We then drive the same hierarchical ESN models with S and S', and measure the Euclidean distance between the echo states generated by S' and S. Formally, we measure the difference between $\mathbf{x}^{(i)}_{S'}(t)$ and the original echo states $\mathbf{x}^{(i)}_{S}(t)$ in i-th reservoir layer, which we denote as $\triangle\mathbf{x}^{(i)}(t)$. In this way, we see how long the perturbation affects each layer, as a measure of the "memory span" of the network.

Figs. 7 and 8 present the visualization results of multiscale dynamics on MGS and NARMA, respectively. As a reference model, we also plot the dynamics of a single-layer ESN. In these plots, the red line denotes the single-layer reference ESN with the same number of output weights, and blue lines denote the perturbation effects at each layer. Darker colors correspond to deeper layers. The first thing to notice is that the perturbation has a bigger effect on the Mackey-Glass chaotic attractor system than on NARMA.

In terms of the different systems, first consider the grouped-ESN: It can be readily seen in the Figures that, since it is simply a collection of shallow ESNs, the grouped-ESN mimics the behavior of the single ESN reference. The MESM, which enforces the Echo-State Property at all layers, displays a smooth response at all layers, with the perturbation diminishing in the deeper layers. However, the time scale of the response among different layers in MESM are nearly identical, showing it does not produce significant multiscale behavior in the hierarchy.

On the other hand, the deepESN, which directly stacks reservoirs with random weights, has wildly different reactions in the MG case versus the NARMA case. In the MG case, it enlarges the added noise in the intermediate layers, but this diminishes in the last layer. Hence, it displays inconsistent multi-scale dynamics, and the deepest layers do not react in this case. For the NARMA system, it has dynamics very similar to a shallow ESN (red line) in the early layers, and again, very little response in later layers.

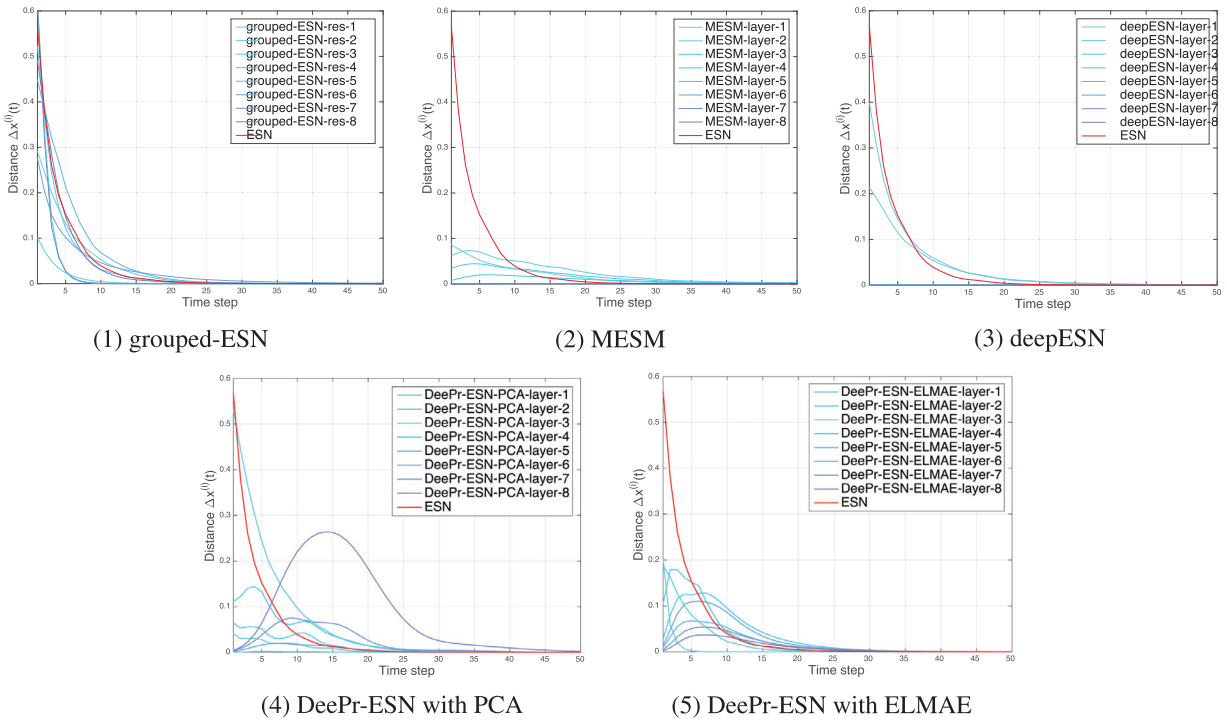


Fig. 8. Visualization of the multiscale dynamics on NARMA one-step ahead direct prediction task. The staring point at x-axis is corresponding to the perturbed time 200.

The DeePr-ESN PCA model shows some multi-scale dynamics, including later layers, but its memory peters out around the 60th time step. On the other hand, the DeePr-ESN with ELMAE shows rich, multiscale dynamics, with longer memory at deeper layers, which is a desirable property. This shows that the deeper layers respond over a longer time scale, as one would expect in a multiscale system. The same is not true in the NARMA case (Fig. 8), where there is little evidence of multiscale dynamics in any case. However, the DeePr-ESNs both display longer memory than the other three systems. As shown in Fig. 8, compared with the results by deepESN, our DeePr-ESN still shows relatively rich multi-scale dynamics in different layers, which is mainly attributed to the design of our multiple projection-encoding hierarchical structure. These results verify that the proposed hierarchical framework generates rich multiscale dynamic behaviors and hence is very suitable for modeling temporal data that require explicit support for multiple time scales and temporal hierarchy.

As another, more quantitative measure of time-scale diversity, we apply two quantitative measures recommended by Gallicchio et al [8] to describe time-scale diversification.

Let $P^{(i)} = argmax_t(\triangle \mathbf{x}^{(i)}(t) > \epsilon)$ (we set ϵ to 10E-4) denote the maximum duration of the perturbation effect on i-th reservoir layer, $\{O^{(i)}\}_{i=1}^{K}$ (K is the number of reservoirs) denote the permutation of $\{1, 2, \ldots, K\}$ by ordering $\{P^{(i)}\}_{i=1}^{K}$. Ideally, ranking $\{O^{(i)}\}_{i=1}^{K}$ should be consistent with the identity permutation ranking $\{1, 2, \ldots, K\}$ with increasing duration of the perturbation effect layer by layer, which means the model can capture multiscale dynamics in the hierarchy. There are two distances [21] between above two rankings: the Kendall's tau, $\mathcal{K}$:

$$\mathcal{K} = |(i_1, i_2) : (1 \le i_1 < i_2 \le K) \wedge (O^{(i_1)} > O^{(i_2)}| \tag{23}$$

and the Spearman's footrule distances, $\mathcal{F}$

$$\mathcal{F} = \sum_{i=1}^{K} \|i - O^{(i)}\| \tag{24}$$

Intuitively, smaller values of these measures mean better quality of the ordering among the time-scales in different reservoir layers.

The values of these measures are given in Tables 5 and 6. From these results, we can see that our DeePr-ESNs provide the richest multiscale dynamics since they have much smaller $\mathcal{F}$ on both datasets, smaller $\mathcal{K}$ on Mackey-Glass, and similar $\mathcal{K}$ on the NARMA dataset. Note that the average NRMSE results roughly track the $\mathcal{F}$ measure.

Table 5
The time-scales diversification measures (smaller values for $\mathcal{K}$ and $\mathcal{F}$ are better) on Mackey-Glass time series.

Methods	$\mathcal{K}\downarrow$	$\mathcal{F}\downarrow$	avg. NRMSE
Grouped-ESN	4	24	3.02E-02
MESM	4	18	1.94E-02
deepESN	3	24	2.37E-02
DeePr-ESN with PCA	1	4	8.14E-03
DeePr-ESN with ELMAE	0	0	9.08E-04

Table 6
The time-scales diversification measures (smaller values for $\mathcal{K}$ and $\mathcal{F}$ are better) on NARMA time series.

Methods	$\mathcal{K}\downarrow$	$\mathcal{F}\downarrow$	avg. NRMSE
Grouped-ESN	3	22	1.27E-01
MESM	3	32	1.31E-01
deepESN	2	24	1.20E-01
DeePr-ESN with PCA	2	12	1.15E-01
DeePr-ESN with ELMAE	3	10	8.20E-02

5. Conclusion

Hierarchical multiscale structure naturally exist in many time series, a phenomenon that is difficult to capture by a conventional ESN. To overcome this limitation, we propose a novel hierarchical reservoir computing framework called Deep Projection-encoding Echo State Network (DeePr-ESN). Instead of directly stacking reservoirs, we combine the randomly-generated reservoirs with unsupervised encoders, retaining the high-dimensional projection capacity as well as the efficient learning of reservoir computing. Through this multiple projection-encoding system, we not only perform robust prediction, but we capture the rich multiscale dynamics in each layer.

Reservoir computing is an efficient method to construct recurrent networks that model dynamical systems. This is in stark contrast to deep learning systems, which require extensive training. The former pursues conciseness and effectiveness, but the latter focuses on the capacity to learn abstract, complex features in the service of the task. Thus, there is a gap between the merits and weaknesses of these two approaches, and a potentially fruitful future direction is to discover a way to bridge these two models, and achieve a balance between the efficiency of one and the feature learning of the other. Our DeePr-ESN provides a novel view towards bridging this gap between reservoir computing and deep learning.

Declaration of Competing Interest

The authors declared that they have no conflicts of interest to this work.

Acknowledgments

The work described in this paper was partially funded by the National Natural Science Foundation of China (Grant No. 61502174, 61872148), the Natural Science Foundation of Guangdong Province (Grant No. 2017A030313355, 2017A030313358), and the Guangzhou Science and Technology Planning Project (Grant No. 201704030051, 201902010020). It was also supported by the National Science Foundation (USA) grant SMA 1041755 to the Temporal Dynamics of Learning Center, an NSF Science of Learning Center.

Appendix A. Analysis of the computational cost

Here, we provide an analysis of DeePr-ESN's computational complexity.

Assuming a DeePr-ESN has K reservoirs and $K-1$ PCA-based encoders, where all reservoirs' sizes are fixed by N, and the reduced dimensionality is M ($M<N$). Given T-length D-dimensional input sequences (let the washout length of each reservoir T_{washout} to be zero), we can analyze the computational complexity of DeePr-ESN as follows.

The complexity at the steps of high-dimensional projection (5) and update (6) in i-th reservoir can be given by

$$\begin{cases} \mathcal{C}_{res(i)} = \mathcal{O}(2\alpha TN^2 + 2TND), \ i = 1, & \text{(A.1)} \\ \mathcal{O}(2\alpha TN^2 + 2TNM), \ i = 2, 3, \ldots, K. & \text{(A.2)} \end{cases}$$

where the sparsity α is small (usually fixed by 0.01).

Besides, the complexity of encoding j-th states with PCA mentioned before can be computed by

$$\mathcal{C}_{enc(j)} = \mathcal{O}(TN^2 + N^2M), \ j = 1, 2, \ldots, K-1. \quad \text{(A.3)}$$

After updating echo states of all the layers at all the time stamps, we can collect the last reservoir states, inputs and all the middle-layer-encoded features into a matrix $\mathbf{M}$ with the size of $(N + (K-1)M + D) \times T$ which is full row rank. In this way, solving the regression problem in (15) has the complexity:

$$\mathcal{C}_{regression} = \mathcal{O}((T + (P/3))P^2, \tag{A.4}$$

where $P = N + (K-1)M + D$. Since the dimension of a reservoir usually is much larger than inputs and encoders' sizes, we can assume that $N \gg M$ and $N \gg D$. And then, we have $P \approx N$. In this way, $\mathcal{C}_{regression}$ can be approximately rewritten as $\mathcal{O}(TN^2 + N^3)$. Further, if T is much larger than N (high dimension property of time series), then we can have $T \gg N$ and $\mathcal{C}_{regression} = \mathcal{O}(TN^2)$.

Finally, the computational complexity of DeePr-ESN can be computed by

$$\mathcal{C}_{\text{DeePr-ESN}} = \sum_{i=1}^{K} \mathcal{C}_{res(i)} + \sum_{j=1}^{K-1} \mathcal{C}_{enc(j)} + \mathcal{C}_{regression} \tag{A.5}$$

That is

$$\mathcal{C}_{\text{DeePr-ESN}} \approx \mathcal{O}(2\alpha TKN^2 + 2TND + (K-1)2TNM + (K-1)(TN^2 + N^2M) + TN^2) \approx \mathcal{O}(KTN^2) \tag{A.6}$$

It can be seen that, with efficient unsupervised encoders (e.g., PCA) and the assumption of high dimension property of time series, the computational complexity of DeePr-ESN is $\mathcal{O}(KTN^2)$. This is the *training* complexity of DeePr-ESN, and the run-time complexity is much smaller.

Besides, a conventional single-reservoir ESN's computational complexity is

$$\mathcal{C}_{\text{ESN}} = \mathcal{C}_{res} + \mathcal{C}_{regression} \approx \mathcal{O}(2\alpha TN^2 + 2TND + +TN^2) \approx \mathcal{O}(TN^2) \tag{A.7}$$

Therefore, our DeePr-ESN can achieve equivalent computational performance to a single-reservoir ESN, which shows that our DeePr-ESN remains the high computational efficiency of traditional reservoir-computing networks.

Appendix B. Main results of hyper-parameters optimized by using GA

Table B1
Detailed information about hyper-parameters of all models on MG 84-step-ahead direct prediction task.

MGS-84	#layer	#outputs	NRMSE		1	2	3	4	5	6	7	8
ESN	1	1200	4.79E-02	*IS*	9.00E-01							
				SR	9.65E-01							
				γ	6.37E-01							
φ-ESN	2	900	8.54E-02	*IS*	7.91E-01	2.20E-02						
				SR	9.70E-01	2.31E-01						
				γ	8.81E-01	2.34E-01						
R^{2}SP	2	2400	2.14E-02	*IS*	9.83E-01							
				SR	8.00E-01							
				γ	3.96E-01							
				IS-ELM(1)	4.12E-01							
				IS-ELM(2)	5.00E-01							
MESM	6	1800	1.55E-02	*IS*	8.18E-01	8.76E-01	7.94E-01	9.66E-01	9.93E-01	8.67E-01		
				SR	9.80E-01	8.94E-01	9.64E-01	9.51E-01	9.26E-01	3.92E-01		
				γ	9.82E-01	2.08E-01	2.57E-01	7.22E-01	6.73E-01	5.89E-01		
deepESN	6	1800	1.71E-02	*IS*	4.53E-01	8.83E-02	2.45E-02	3.13E-03	7.80E-02	7.77E-02		
				SR	4.97E-01	7.70E-01	8.73E-01	9.65E-01	7.56E-01	1.93E-01		
				γ	8.99E-01	3.91E-02	2.88E-01	4.11E-01	7.38E-01	5.92E-01		
grouped-ESN	6	1800	9.73E-03	*IS*	1.00E+00	6.07E-01	9.44E-01	4.33E-01	5.28E-01	7.00E-03		
				SR	1.00E+00	9.92E-01	8.55E-01	9.18E-01	9.97E-01	4.13E-01		
				γ	5.69E-01	5.03E-01	3.65E-01	9.51E-01	7.25E-01	4.48E-01		
DeePr-ESN (PCA, 60)	8	2400	8.14E-03	*IS*	5.34E-01	8.86E-01	5.30E-01	7.13E-01	2.88E-02	6.72E-02	3.65E-01	5.00E-02
				SR	7.84E-01	9.44E-01	2.10E-01	5.61E-01	8.04E-01	5.58E-01	2.51E-01	6.56E-01
				γ	2.63E-01	6.04E-01	9.55E-02	5.72E-01	4.02E-01	9.29E-01	3.92E-01	2.04E-01
DeePr-ESN (ELMAE, 60)	8	2400	9.08E-04	*IS*	9.55E-01	9.34E-01	7.90E-01	9.86E-01	9.90E-01	7.51E-01	8.81E-01	6.48E-01
				SR	9.67E-01	1.00E+00	7.78E-01	9.32E-01	8.64E-01	9.30E-01	9.81E-01	2.58E-01
				γ	3.04E-01	5.14E-01	6.55E-02	1.10E-01	8.65E-02	4.47E-01	8.67E-02	7.97E-01

Table B2
Detailed information about hyper-parameters of all models on NARMA one-step-ahead direct prediction task.

NARMA	#layer	#outputs	NRMSE		1	2	3	4	5	6	7	8
ESN	1	1500	1.24E-01	*IS*	9.89E-02							
				SR	9.04E-01							
				γ	9.60E-01							
φ-ESN	1	1800	1.31E-01	*IS*	1.66E-01	5.89E-03						
				SR	9.53E-01	5.71E-01						
				γ	9.94E-01	2.57E-01						
R^2SP	2	1500	1.31E-01	*IS*	1.28E-01							
				SR	9.31E-01							
				γ	9.46E-01							
				IS-ELM(1)	5.97E-01							
				IS-ELM(2)	5.26E-01							
MESM	6	1800	1.17E-01	*IS*	9.33E-02	9.92E-01	9.52E-01	8.42E-01	9.20E-01	4.77E-01		
				SR	9.67E-01	8.48E-01	2.19E-02	9.36E-01	2.15E-01	4.61E-01		
				γ	9.90E-01	9.61E-01	8.69E-01	7.61E-01	3.35E-01	6.19E-01		
deepESN	7	2100	1.05E-01	*IS*	2.11E-01	4.36E-02	2.85E-01	8.64E-01	3.66E-01	7.02E-02	9.66E-01	
				SR	9.45E-01	7.94E-01	7.70E-01	2.39E-01	1.26E-01	7.28E-02	6.01E-01	
				γ	9.84E-01	3.51E-01	4.72E-01	3.54E-01	2.85E-01	8.82E-01	3.38E-01	
grouped-ESN	8	2400	1.27E-01	*IS*	2.86E-01	3.08E-01	6.67E-01	7.19E-01	4.51E-01	7.39E-01	8.29E-01	5.19E-01
				SR	8.71E-01	2.46E-01	8.14E-01	4.98E-01	8.86E-01	8.29E-01	3.73E-01	8.97E-01
				γ	9.88E-01	3.27E-01	8.21E-01	9.09E-01	9.89E-01	3.70E-01	7.87E-01	9.46E-01
DeePr-ESN (PCA, 300)	7	2100	1.02E-01	*IS*	5.33E-01	3.22E-02	2.32E-01	1.76E-01	1.60E-03	6.16E-01	3.46E-01	
				SR	8.91E-01	3.80E-01	3.67E-01	7.20E-01	3.45E-02	7.61E-01	5.66E-01	
				γ	9.80E-01	2.24E-01	4.76E-01	8.52E-01	4.11E-01	9.26E-01	9.18E-01	
DeePr-ESN (ELMAE, 200)	7	2100	6.93E-02	*IS*	2.17E-01	8.38E-01	9.43E-01	5.87E-01	1.64E-01	7.74E-01	2.09E-01	
				SR	3.05E-01	8.67E-01	7.15E-02	2.27E-01	1.70E-01	9.11E-02	4.70E-01	
				γ	8.95E-01	9.54E-01	3.86E-01	1.69E-01	8.32E-01	8.20E-01	6.54E-01	

Table B3
Detailed information about hyper-parameters of all models on Sunspot time series one-step ahead direct prediction task.

Sunspot	#layer	#outputs	NRMSE		1	2	3	4
ESN	1	1200	3.18E-02	*IS*	1.35E-01			
				SR	8.25E-01			
				γ	8.98E-01			
φ-ESN	1	900	2.42E-02	*IS*	3.54E-01	2.09E-02		
				SR	8.56E-01	3.55E-02		
				γ	9.55E-01	9.52E-01		
R^2SP	2	1200	5.19E-02	*IS*	1.30E-02			
				SR	8.90E-01			
				γ	9.61E-01			
				IS-ELM(1)	7.12E-01			
				IS-ELM(2)	9.87E-01			
MESM	2	600	1.95E-02	*IS*	6.48E-01	4.74E-01		
				SR	5.69E-01	6.59E-01		
				γ	9.82E-01	9.08E-01		
deepESN	2	600	2.02E-02	*IS*	2.51E-01	9.69E-02		
				SR	8.60E-01	1.05E-01		
				γ	8.14E-01	5.79E-01		
Grouped-ESN	4	1200	4.78E-02	*IS*	2.38E-02	4.50E-01	9.06E-01	5.20E-03
				SR	2.80E-01	1.19E-01	5.97E-04	7.99E-01
				γ	9.39E-01	7.01E-01	6.36E-01	9.25E-01
DeePr-ESN (PCA, 120)	2	600	1.90E-02	*IS*	4.28E-01	2.67E-02		
				SR	7.22E-01	2.29E-01		
				γ	9.92E-01	9.54E-01		
DeePr-ESN (ELMAE, 120)	7	2100	1.89E-02	*IS*	1.52E-03	3.91E-02		
				SR	7.88E-01	4.45E-01		
				γ	9.89E-01	8.92E-01		

Table B4
Detailed information about hyper-parameters of all models on Temperature time series one-step ahead direct prediction task.

Temperature	#layer	#outputs	NRMSE		1	2	3
ESN	1	1200	1.53E-01	*IS*	6.89E-02		
				SR	3.68E-01		
				γ	1.00E+00		
φ-ESN	2	900	1.51E-01	*IS*	4.03E-04	7.76E-02	
				SR	9.22E-01	9.01E-01	
				γ	9.69E-01	3.31E-01	
R^2SP	2	900	1.56E-01	*IS*	1.42E-02		
				SR	5.22E-01		
				γ	8.24E-01		
				IS-ELM(1)	8.67E-01		
				IS-ELM(2)	1.86E-01		
MESM	2	600	1.28E-01	*IS*	2.10E-02	1.09E-01	
				SR	1.90E-01	9.05E-01	
				γ	8.59E-01	9.17E-01	
deepESN	2	600	1.28E-01	*IS*	3.77E-02	7.29E-01	
				SR	5.70E-01	2.28E-01	
				γ	9.85E-01	6.73E-01	
Grouped-ESN	3	900	1.56E-01	*IS*	1.48E-01	6.01E-01	1.03E-02
				SR	3.96E-02	9.91E-02	5.51E-01
				γ	9.10E-01	9.36E-01	9.29E-01
DeePr-ESN (PCA, 90)	2	600	1.26E-01	*IS*	7.16E-03	1.18E-01	
				SR	7.26E-01	4.02E-01	
				γ	9.58E-01	7.16E-01	
DeePr-ESN (ELMAE, 200)	7	2100	1.27E-01	*IS*	1.52E-03	3.91E-02	
				SR	7.88E-01	4.45E-01	
				γ	9.89E-01	8.92E-01	

References

[1] R.J. Bray, R.E. Loughhead, Sunspots, Dover Publications, 1979.
[2] J. Butcher, D. Verstraeten, B. Schrauwen, C. Day, P. Haycock, Reservoir computing and extreme learning machines for non-linear time-series data analysis, Neural Netw. 38 (2013) 76–89, doi:10.1016/j.neunet.2012.11.011.
[3] J. Butcher, D. Verstraeten, B. Schrauwen, C. Day, P. Haycock, Extending reservoir computing with random static projections: a hybrid between extreme learning and rc, in: Esann 2010, European Symposium on Artificial Neural Networks, Bruges, Belgium, April 28–30, 2010, Proceedings, 2010, pp. 303–308.
[4] E. Cambria, G.B. Huang, L.L.C. Kasun, H. Zhou, C.M. Vong, J. Lin, J. Yin, Z. Cai, Q. Liu, K. Li, V.C.M. Leung, L. Feng, Y.S. Ong, M.H. Lim, A. Akusok, A. Lendasse, F. Corona, R. Nian, Y. Miche, P. Gastaldo, R. Zunino, S. Decherchi, X. Yang, K. Mao, B.S. Oh, J. Jeon, K.A. Toh, A.B.J. Teoh, J. Kim, H. Yu, Y. Chen, J. Liu, Extreme learning machines [trends controversies], IEEE Intell. Syst. 28 (6) (2013) 30–59.
[5] J. Chung, S. Ahn, Y. Bengio, Hierarchical multiscale recurrent neural networks (2016). arXiv: 1609.01704.
[6] S. Fernández, A. Graves, J. Schmidhuber, Sequence labelling in structured domains with hierarchical recurrent neural networks, in: IJCAI, 2007, pp. 774–779.
[7] C. Gallicchio, A. Micheli, Architectural and markovian factors of echo state networks, Neural Netw. 24 (5) (2011) 440–456, doi:10.1016/j.neunet.2011.02.002.
[8] C. Gallicchio, A. Micheli, Deep reservoir computing: a critical analysis, European Symposium on Artificial Neural Networks, 2016.
[9] C. Gallicchio, A. Micheli, Echo state property of deep reservoir computing networks, Cogniti. Comput. (6) (2017) 1–14.
[10] C. Gallicchio, A. Micheli, L. Pedrelli, Deep reservoir computing: a critical experimental analysis, Neurocomputing (2017).
[11] A. Graves, A.-r. Mohamed, G. Hinton, Speech recognition with deep recurrent neural networks, in: Acoustics, speech and signal processing (icassp), 2013 ieee international conference on, IEEE, 2013, pp. 6645–6649.
[12] M. Han, M. Xu, Laplacian echo state network for multivariate time series prediction, IEEE Trans. Neural Netw. Learn.Syst. 29 (1) (2018) 238–244, doi:10.1109/TNNLS.2016.2574963.
[13] M. Hermans, B. Schrauwen, Recurrent kernel machines: computing with infinite echo state networks, Neural Comput. 24 (1) (2012) 104–133.
[14] M. Hermans, B. Schrauwen, Training and analysing deep recurrent neural networks, in: C. Burges, L. Bottou, M. Welling, Z. Ghahramani, K. Weinberger (Eds.), Advances in Neural Information Processing Systems 26, 2013, pp. 190–198.
[15] G. Huang, G.-B. Huang, S. Song, K. You, Trends in extreme learning machines: a review, Neural Netw. 61 (2015) 32–48, doi:10.1016/j.neunet.2014.10.001.
[16] H. Jaeger, H. Haas, Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication, Science 304 (5667) (2004) 78–80.
[17] H. Jaeger, Short term memory in echo state networks, GMD Report, 2002.
[18] H. Jaeger, Discovering multiscale dynamical features with hierarchical echo state networks, Vtls Inc 35 (2) (2007) 277–284.
[19] H. Jaeger, Erratum note for the techreport, the “echo state” approach to analysing and training recurrent neural networks, Technical report, 2010.
[20] H. Jaeger, M. Lukoševičius, D. Popovici, U. Siewert, Optimization and applications of echo state networks with leaky-integrator neurons, Neural Netw. 20 (3) (2007) 335–352. Echo State Networks and Liquid State Machines. 10.1016/j.neunet.2007.04.016.
[21] R. Kumar, S. Vassilvitskii, Generalized distances between rankings, in: Proceedings of the 19th International Conference on World Wide Web, in: WWW '10, ACM, New York, NY, USA, 2010, pp. 571–580, doi:10.1145/1772690.1772749.
[22] M. Lukoševičius, H. Jaeger, B. Schrauwen, Reservoir computing trends, KI - Künstliche Intelligenz 26 (4) (2012) 365–371, doi:10.1007/s13218-012-0204-5.
[23] M. Lukoševičius, H. Jaeger, Survey: reservoir computing approaches to recurrent neural network training, Comput. Sci. Rev. 3 (3) (2009) 127–149, doi:10.1016/j.cosrev.2009.03.005.

[24] Q. Ma, W. Chen, J. Wei, Z. Yu, Direct model of memory properties and the linear reservoir topologies in echo state networks, Appl. Soft Comput. 22 (2014) 622–628, doi:10.1016/j.asoc.2014.04.038.
[25] W. Maass, T. Natschläger, H. Markram, Real-time computing without stable states: a new framework for neural computation based on perturbations, Neural Comput. 14 (11) (2002) 2531–2560, doi:10.1162/089976602760407955.
[26] L.J.P.V.D. Maaten, E.O. Postma, H.J.V.D. Herik, Dimensionality reduction: a comparative review, J. Mach. Learn. Res. 10 (1) (2009).
[27] Z.K. Malik, A. Hussain, Q.J. Wu, Multilayered echo state machine: a novel architecture and algorithm, IEEE Trans. Cybernet. 47 (4) (2017) 946–959.
[28] M. Massar, S. Massar, Mean-field theory of echo state networks, Phys. Rev. E, Stat. NonlinearSoft Matter Phys. 87 (4) (2013) 042809, doi:10.1103/physreve.87.042809.
[29] M. Mitchell, An Introduction to Genetic Algorithms, MIT Press, Cambridge, MA, USA, 1996.
[30] R. Pascanu, C. Gulcehre, K. Cho, Y. Bengio, How to construct deep recurrent neural networks (2013) arXiv:1312.6026.
[31] J. Qiao, F. Li, H. Han, W. Li, Growing echo-state network with multiple subreservoirs, IEEE Trans. Neural Netw. Learn.Syst. 28 (2) (2017) 391–404, doi:10.1109/tnnls.2016.2514275.
[32] A. Rodan, P. Tino, Minimum complexity echo state network, IEEE Trans. Neural Netw. 22 (1) (2011) 131–144, doi:10.1109/TNN.2010.2089641.
[33] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Nature 323 (6088) (1986) 533–536.
[34] A. Sharma, K.K. Paliwal, Fast principal component analysis using fixed-point algorithm, Pattern Recognit. Lett. 28 (10) (2007) 1151–1155, doi:10.1016/j.patrec.2007.01.012.
[35] C. Shen, C. Liu, H. Tan, Z. Wang, D. Xu, X. Su, Hybrid-augmented device fingerprinting for intrusion detection in industrial control system networks, IEEE Wirel. Commun. 25 (6) (2018) 26–31, doi:10.1109/MWC.2017.1800132.
[36] Z. Shi, M. Han, Support vector echo-state machine for chaotic time-series prediction, Trans. Neur. Netw. 18 (2) (2007) 359–372, doi:10.1109/TNN.2006.885113.
[37] SILSO World Data Center, The international sunspot number, Int. Sunspot Number Monthly Bull.Online Catalogue (1749-2016).
[38] H. Soh, Y. Demiris, Spatio-temporal learning with the online finite and infinite echo-state gaussian processes, IEEE Trans. Neural Netw. Learn.Syst. 26 (3) (2015) 522–536, doi:10.1109/TNNLS.2014.2316291.
[39] A. Sorjamaa, J. Hao, N. Reyhani, Y. Ji, A. Lendasse, Methodology for long-term prediction of time series, Neurocomputing 70 (16) (2007) 2861–2869. Neural Network Applications in Electrical Engineering Selected papers from the 3rd International Work-Conference on Artificial Neural Networks (IWANN 2005). 10.1016/j.neucom.2006.06.015.
[40] J. Tang, C. Deng, G.B. Huang, Extreme learning machine for multilayer perceptron, IEEE Trans. Neural Netw. Learn.Syst. 27 (4) (2016) 809–821, doi:10.1109/TNNLS.2015.2424995.
[41] A.N. Tikhonov, V.Y. Arsenin, Solutions of ill-posed problems, Math. Comput. 32 (144) (1977). 491–491
[42] Time Series Data Library, Daily minimum temperatures in melbourne, australia, 1981–1990.
[43] F. Triefenbach, A. Jalalvand, K. Demuynck, J.P. Martens, Acoustic modeling with hierarchical reservoirs, IEEE Trans. Audio Speech Lang. Process. 21 (11) (2013) 2439–2450.
[44] F. Triefenbach, K. Demuynck, J.P. Martens, Large vocabulary continuous speech recognition with reservoir-based acoustic models, IEEE Signal Process. Lett. 21 (3) (2014) 311–315.
[45] F. Triefenbach, A. Jalalvand, B. Schrauwen, J.-P. Martens, Phoneme recognition with large hierarchical reservoirs, in: Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 2, in: NIPS'10, Curran Associates Inc., USA, 2010, pp. 2307–2315.
[46] D. Verstraeten, B. Schrauwen, M. D'Haene, D. Stroobandt, 2007 Special issue: an experimental unification of reservoir computing methods, Neural Netw. 20 (3) (2007) 391–403, doi:10.1016/j.neunet.2007.04.003.
[47] G. Wainrib, M.N. Galtier, A local echo state property through the largest lyapunov exponent, Neural Netw. 76 (2016) 39–45, doi:10.1016/j.neunet.2015.12.013.
[48] Y. Xia, B. Jelfs, M.M.V. Hulle, J.C. Principe, D.P. Mandic, An augmented echo state network for nonlinear adaptive filtering of complex noncircular signals, IEEE Trans. Neural Netw. 22 (1) (2011) 74–83, doi:10.1109/TNN.2010.2085444.
[49] M. Xu, M. Han, Adaptive elastic echo state network for multivariate time series prediction, IEEE Trans. Cybernet. 46 (10) (2016) 2173–2183.
[50] I.B. Yildiz, H. Jaeger, S.J. Kiebel, Re-visiting the echo state property, Neural Netw. 35 (2012) 1–9, doi:10.1016/j.neunet.2012.07.005.