THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A
Volume 24, Number 2/2023, pp. 179–190
DOI:

DUAL-KERNEL ECHO STATE NETWORK FOR NONLINEAR TIME SERIES PREDICTION

Guoxin TANG[1], Lang YU[2], Wangyong LV[1], Yuhuai SUN[1]

[1] Sichuan Normal University, College of Mathematical Science, Chengdu, 610066, China
[2] Chongqing University, College of Mathematics and Statistics, Chongqing, 401331, China
Corresponding author: Yuhuai SUN, E-mail: syh1997@sicnu.edu.cn

Abstract. An echo state network (ESN) is a recurrent neural network (RNN) often applied to nonlinear time series prediction. The traditional ESN randomly generates the weights of the input layer and the reservoir layer and does not change these weights, and generally only learns the weights of the output layer through linear regression, so the training speed is very fast. In this work, we propose a novel kernel echo state network (KESN). In KESN, the random weights of the input layer are removed and a series of gaussian kernels are used to replace the neurons in the input layer. Similar to radial basis function (RBF) neural networks, KESN can use the k-means algorithm to generate the kernel center and estimate the bandwidth of the kernel function. We prove that a KESN has echo state property, which is an important factor of KESN that can normally work. Furthermore, kernel ridge regression (KRR) is used to learn the weights of the output layer instead of a simple linear model. Finally, to obtain the optimal parameters of the model, the tree-structured parzen estimator approach (TPE) is used to optimize the hyperparameters of the model. In a time series prediction experiment, it is proved that KESN is more stable and performs better than the echo state network which randomly generates weights and trains output weights using linear models. We found that the reservoir layer weights are equivalent to a dropout operation, and the KESN is inherently equivalent to a regularized neural network. We call the KRR-based KESN dual-kernel echo state network (DKESN).

Key words: recurrent neural network, Kernel echo state network, Kernel ridge regression, time series prediction, tree-structured Parzen estimator.

1. INTRODUCTION

A time series is described as a sequence of observations from a system. Most of the time these observations are nonlinear. Many time series in the real world such as weather, stocks and chaotic systems can be regarded as a complex nonlinear time series. It is very important to accurately forecast time series, which can guide us to formulate relevant policies or make correct decisions.

In order to model nonlinear systems, traditional mathematical methods such as the grey model [1, 2] and support vector machines [3] are widely used. However, these models may not be applicable if the nonlinear system has complex properties such as solid nonlinearity and multivariate coupling [4]. Artificial neural networks are often used for nonlinear modelings, such as image classification, speech recognition, etc [5]. The echo state network (ESN [6]) is a recurrent neural network (RNN) whose structure is consistent with the classical RNN. However, the weights of the input layer and the hidden layer (also called the reservoir layer) of ESN are randomly generated matrices, and only the weights of the output layer need to be trained. Generally speaking, the weights of the output layer are usually obtained by linear least squares. Hence, compared with classical RNN, ESN has a faster training speed and is less prone to gradient explosion or gradient disappearance [7]. Hence, ESN and its variants have been widely used in time series forecasting and nonlinear system modeling. Figure 1(a) shows the structure of a classical ESN.

Although ESN has many advantages, there are still some problems hindering its development. Since the reservoir layer of ESN is randomly generated, the weight matrix of the random layer is generally required to be very large and sparse, which may cause the state matrix $\mathrm{S} = [\boldsymbol{s}_1, \boldsymbol{s}_2, ..., \boldsymbol{s}_\tau]$ in Figure 1 to not be full rank. Hence, using the conventional least squares method to train ESN may cause ill-posed problem and lead to degradation of the model performance. To avoid the ill-posed problem, the regularization method is an effective means. Among them, Jacobs et al. add ℓ_2 regularization to ESN [8]. Qiao et al. employed an adaptive ℓ_1 regularization method to solve the possible collinearity problem in ESN and obtain the sparse structure of the model [9]. Xu proposed an ESN for elastic net regression, which combines the advantages of ℓ_1 regularization and ℓ_2 regularization [10]. Wang et al. proposed a sparse bayesian method to train ESN, which can evaluate the probability distribution of output values instead of fitting the data, thus greatly eliminating the ill-posed problem [11]. Furthermore, Qiao et al. proposed a sparse recursive least squares online ESN algorithm to avoid the overfitting problem [12]. Although these studies address the ill-posed problem by imposing a regularization term on the ESN, the algorithm for training the ESN is a linear model. Meanwhile, these studies have put the focus on getting a sparser model. However, a compressed model not only makes the solution of the model difficult but may also lead to a decrease in the nonlinear capability of the model. Yao et al. extended the traditional ESN to a fractional-order ESN (FESN) [13] to further improve the memory performance of the ESN, and trained the FESN using gradient descent method.

For ESN, the network structure and training algorithm are two important factors that affect its performance. However, most of these works have studied the learning algorithms of ESN, and there are relatively few studies on the structure of ESN. Hu et al. extend the ESN to deep ESN (DeepESN) by stacking multiple reservoir layers [14]. Patrick et al. proposed a quadratic echo state network (QESN), which requires learning two output matrices instead of one [15]. DeepESN and QESN learn the output layer weights using least squares and ridge regression, respectively.

In our study, we note that in existing studies, the weights of the input and reservoir layers of the ESN are randomly generated. Only the weights of the output layer need to be learned. Hence, random weights introduce more uncertainty in addition to the ill-posed problem, which may also lead to model performance degradation. Hence, for an input layer with K neurons, we generate the K centers of the original sequence through the k-means algorithm, then apply a kernel function to map the original data into a high-dimensional space. Finally, we use nonlinear kernel ridge regression (KRR, [16]) to learn the weights of ESN instead of the classical linear method. We refer to this ESN as the dual-kernel echo state network (DKESN).

Figure 1 shows the difference between original ESN, DeepESN, QESN and DKESN. We call the part of the basic ESN that does not need to be trained as an ESN unit, and the algorithm for solving the ESN is called a solver. It can be found that DKESN does not have the input layer matrix U, and instead a kernel function map $\mathscr{K}$. For QESN, two output weights W_1 and W_2 need to be learned. The basic structure of DeepESN is the connection of many ESN units, which can have many different deep structures [14, 17, 18]. For the solver, it can be any linear or nonlinear model such as ridge regression, Lasso [19], elastic-net regression [20], huber regression [21], bayesian regression [22], quantile regression [23], and KRR, etc. In addition, the gradient descent method can also be employed to train ESN.

Modeling with DKESN may have the following benefits:

- The input layer no longer uses randomly generated weights, but directly maps the original data through the k-means algorithm and kernel function. This mapping is deterministic and controllable.
- When the data shows strong nonlinearity, the accuracy of ordinary regression models will decrease, and DKESN can better solve this problem.
- Since DKESN only randomly generates the weight matrix of the reservoir layer, this will improve the stability of the model.
- The reservoir matrix is used as a sparse matrix, which will produce a dropout effect [24], which is beneficial to improve the generalization ability of the model.

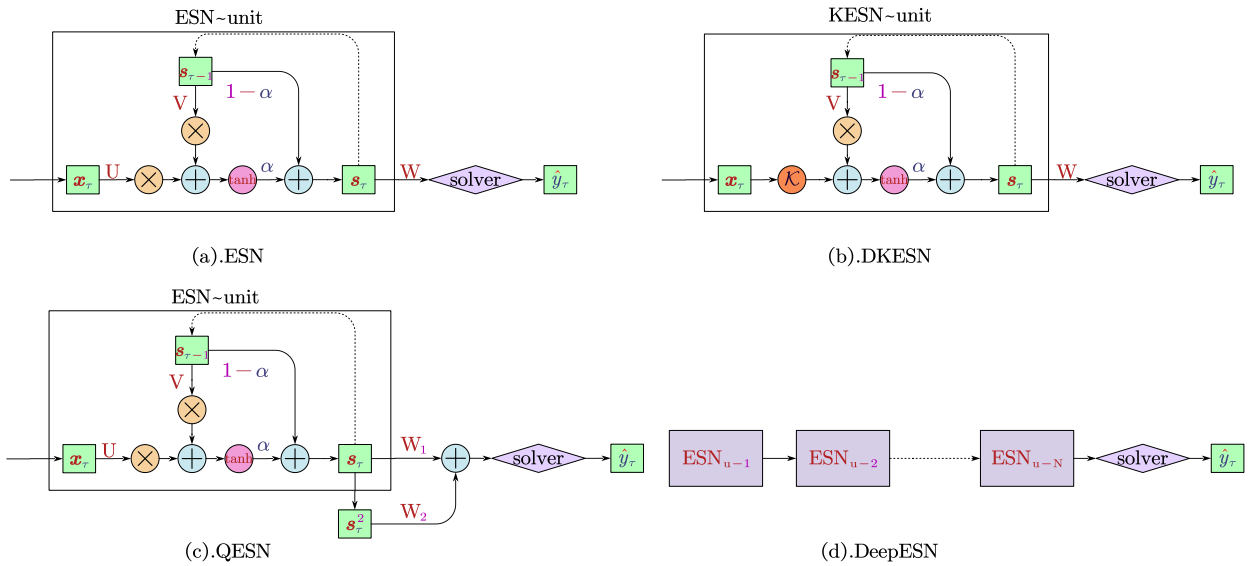

Fig. 1 – Four ESN structures: $\times$ represents matrix multiplication, $+$ represents matrix addition, tanh represents activation function, and $\mathscr{K}$ represents kernel function mapping, and solver represents learning algorithm.

Furthermore, we also found in our experiments that performing feature compression and feature selection on the model will degrade the performance of the model, and we cannot add regularization to the model excessively.

The remainder of this paper is organized as follows. The basic echo state network is introduced in Section 2. In Section 3, the modeling method of DKESN input layer is given, and the echo state characteristic of DKESN is proved. Finally, the hyperparameter optimization method of DKESN is given. In Section 5, a case study is given to demonstrate the advantages of DKESN over other ESN models. Conclusions and prospects are given in Section 5.

2. PRELIMINARY: THE ECHO STATE NETWORK

Let $\mathscr{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_t]^{\mathrm{T}}$ and $\mathscr{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_t]^{\mathrm{T}}$ be the input and output of a nonlinear system, respectively. An ESN with a leak rate α can be represented as:

$$\tilde{\boldsymbol{s}}_\tau = \tanh\left(\mathrm{U}\boldsymbol{x}_\tau + \mathrm{V}\boldsymbol{s}_{\tau-1}\right), \tag{1}$$

$$\boldsymbol{s}_\tau = \alpha\tilde{\boldsymbol{s}}_\tau + (1-\alpha)\,\boldsymbol{s}_{\tau-1}, \tag{2}$$

$$\hat{\boldsymbol{y}}_\tau = \mathrm{W}^{\mathrm{T}}\begin{bmatrix}\boldsymbol{x}_\tau \\ \boldsymbol{s}_\tau\end{bmatrix}, \tag{3}$$

where $\boldsymbol{x}_\tau \in \mathbb{R}^K$ and $\boldsymbol{y}_\tau \in \mathbb{R}^L$, $\mathrm{U} \in \mathbb{R}^{N\times K}$, $\mathrm{V} \in \mathbb{R}^{N\times N}$ and $\mathrm{W} \in \mathbb{R}^{(N+K)\times L}$ are the weight matrices of input layer, reservoir layer and output layer, respectively.

In ESN, only W is a parameter that needs to be trained. Unless otherwise specified, the formulas in this paper will set $L = 1$. For convenience, let:

$$\mathrm{S} = \begin{bmatrix}\boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_t \\ \boldsymbol{s}_1 & \boldsymbol{s}_1 & & \boldsymbol{s}_t\end{bmatrix} = \begin{bmatrix}\boldsymbol{z}_1 & \boldsymbol{z}_2 & \cdots & \boldsymbol{z}_t\end{bmatrix}. \tag{4}$$

Hence,

$$\hat{\mathscr{Y}} = \mathrm{W}^{\mathrm{T}}\mathrm{S}. \tag{5}$$

In general, Let the spectral radius of V is $\rho(\mathrm{V})$, we require $\rho(\mathrm{V}) < 1$ to ensure the echo state property. After generating a random matrix V, scale $\rho(\mathrm{V})$ to a suitable value ρ_r in the following way:

$$\mathrm{V}^* = \frac{\rho_r}{\rho(\mathrm{V})}\mathrm{V}. \tag{6}$$

Let $\mathrm{V} = \mathrm{V}^*$, then $\rho(\mathrm{V}) = \rho_r$. It is worth noting that $\rho(\mathrm{V}) < 1$ is only a necessary condition, and sometimes the best results may be obtained when $\rho(\mathrm{V}) \geqslant 1$.

In some studies, Eq.(1) is also written as:

$$\tilde{\boldsymbol{s}}_\tau = \tanh\left(\mathrm{U}\boldsymbol{x}_\tau + \mathrm{V}\boldsymbol{s}_{\tau-1} + \mathrm{F}\boldsymbol{y}_{\tau-1}\right), \tag{7}$$

where $\mathrm{F} \in \mathbb{R}^{N\times L}$ is feedback matrix. In this study, we do not consider ESN with output feedback. The general solution to the ESN is as follows:

$$\hat{\mathrm{W}} = \underset{\mathrm{W}}{arg\min}\frac{1}{2}\left\|\mathscr{Y} - \mathrm{W}^T\mathrm{S}\right\|_2^2 + \mathscr{F}(\mathrm{W}), \tag{8}$$

where $\mathscr{F}(\mathrm{W})$ can be any regularization penalty. Common regularization methods include:

- ℓ_0 regularization: $\mathscr{F} = \lambda\ell_0$, where $\ell_0 = \sum_{i=1}^{N} 1_{\mathrm{W}_i\neq 0}$.
- $\ell_{1/2}$ regularization: $\mathscr{F} = \lambda\ell_{1/2}$, where $\ell_1 = \|\mathrm{W}\|_{1/2}$.
- Lasso: $\mathscr{F} = \lambda\ell_1$, where $\ell_1 = \|\mathrm{W}\|_1$.
- Ridge: $\mathscr{F} = \lambda\ell_2$, where $\ell_2 = \|\mathrm{W}\|_2^2$.
- Elastic-Net: $\mathscr{F} = \lambda(\mu\ell_1 + (1-\mu)\ell_2)$.

Generally speaking, ℓ_0 regularization is an NP-hard problem, and the cost of solving it is very large. The $\ell_{1/2}$ regularization proposed by Xu et al. may be an alternative to ℓ_0 [25]. $\ell_{1/2}$ can produce sparser solutions than ℓ_1, but $\ell_{1/2}$ is solved much faster than ℓ_0. Table 1 shows the methods of solving the model when different regularization terms are used for Eq.(8).

Table 1

Types of ESN

Model name	Structure	$\mathscr{F}$	Method	Reference
ESN	ESN	-	LS	[6]
SBESN	ESN	Sparse Bayesian	ML	[11]
ALESN	ESN	Adaptive ℓ_1	LARS	[9]
DeepESN	DeepESN	-	LS	[14]
OSESN-SRLS	ESN	ℓ_0 or ℓ_1	SRLS	[12]
FESN	FESN	-	GD	[13]
GESN	GESN	-	LS	[26]
QESN	QESN	ℓ_2	LS	[15]
EESN	ESN	$\ell_1 + \ell_2$	LARS	[10]

LS: Least Squares, ML: Maximum Likelihood Estimation, LARS: Least Angle Regression, SRLS: Sparse Recursive Least Squares, GD: Gradient Descent.

A complete computation process of ESN based on ridge regression (RESN) is presented in Algorithm 1.

Algorithm 1: RESN

Input:
Train set: $\mathscr{X}_{\text{train}} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_t]^{\mathrm{T}}, \mathscr{Y}_{\text{rain}} = [\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_t]^{\mathrm{T}}$;
Test set: $\mathscr{X}_{\text{test}} = [\boldsymbol{x}_{t+1}, \boldsymbol{x}_{t+2}, ..., \boldsymbol{x}_{t+k}]^{\mathrm{T}}$;
reservoir size: N;
leak rate: α;
spectral radius: ρ_r ;
penalty coefficient: λ;
Output: $\hat{\boldsymbol{y}} = \text{RESN}(\boldsymbol{x})$

$\mathrm{U} \leftarrow$ random matrix $\in \mathbb{R}^{N \times K}$;
$\mathrm{V} \leftarrow$ random matrix $\in \mathbb{R}^{N \times N}$;
$\mathrm{V} \leftarrow \frac{\rho_r}{\rho(\mathrm{V})}\mathrm{V}$; // Eq.(6)
$\boldsymbol{s}_0 \leftarrow 0$;
for $\tau = 1$ *to* t **do**
  $\tilde{\boldsymbol{s}}_\tau \leftarrow \tanh(\mathrm{U}\boldsymbol{x}_\tau + \mathrm{V}\boldsymbol{s}_{\tau-1})$; // Eq.(1)
  $\boldsymbol{s}_\tau \leftarrow \alpha\tilde{\boldsymbol{s}}_\tau + (1-\alpha)\boldsymbol{s}_{\tau-1}$; // Eq.(2)
end
$\mathrm{S} \leftarrow \begin{bmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_t \\ \boldsymbol{s}_1 & \boldsymbol{s}_1 & & \boldsymbol{s}_t \end{bmatrix}$; // Eq.(4)
$\hat{\mathrm{W}} \leftarrow \underset{\mathrm{W}}{arg\min} \frac{1}{2}\left\|\mathscr{Y} - \mathrm{W}^{\mathrm{T}}\mathrm{S}\right\|_2^2 + \lambda\left\|\mathrm{W}\right\|_2^2$; // Eq.(8)
for $\tau = t+1$ *to* $t+k$ **do**
  $\tilde{\boldsymbol{s}}_\tau \leftarrow \tanh(\mathrm{U}\boldsymbol{x}_\tau + \mathrm{V}\boldsymbol{s}_{\tau-1})$; // Eq.(1)
  $\boldsymbol{s}_\tau \leftarrow \alpha\tilde{\boldsymbol{s}}_\tau + (1-\alpha)\boldsymbol{s}_{\tau-1}$; // Eq.(2)
  $\hat{\boldsymbol{y}}_{\tau-t} \leftarrow \hat{\mathrm{W}}^{\mathrm{T}} \begin{bmatrix} \boldsymbol{x}(\tau) \\ \boldsymbol{s}(\tau) \end{bmatrix}$;
end

3. DUAL-KERNEL ECHO STATE NETWORK AND REGULARIZATION

In this section, we introduce the dual-kernel echo state network and the regularization method for the model. DKESN consists of two parts: kernel echo state network (KESN) and kernel ridge regression(KRR). Furthermore, tree-structured parzen estimator approach(TPE, [27]) is used to optimize the hyperparameters of DKESN.

3.1. Structure: kernel echo state network

Generally speaking, each layer of the neural network plays the role of feature extraction. In order to convert linearly inseparable to linearly separable, SVM and RBF neural networks [28] apply kernel functions to map raw data into high-dimensional space. However, the input layer of ESN is a randomly generated weight matrix, and no activation function is applied either. In other words, the input layer just does a random feature extraction on the original input. An inappropriate matrix may lead to insufficient or counterproductive feature extraction from the original data. Hence, it is a more reasonable choice to use a kernel function instead of the random weight matrix of the input layer.

The general expression of the gaussian kernel function is:

$$\mathscr{K}(\boldsymbol{x}_1, \boldsymbol{x}_2; \sigma) = \exp\left\{-\frac{\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2}{2\sigma^2}\right\}, \tag{9}$$

where σ is the kernel function bandwidth. Similar to the RBF neural network, given the reservoir size N and the input data set $\mathscr{X}$, we need to find the N centers of $\mathscr{X}$. This process is an unsupervised learning and

can be done by any clustering algorithm such as k-means algorithm. Assuming that $\mathscr{X}$ has t samples. Let $\mathscr{C}_{\mathscr{X}} = [\boldsymbol{c}_1, \boldsymbol{c}_2, ..., \boldsymbol{c}_N]$ is the N centers of $\mathscr{X}$. $\mathscr{C}_{\mathscr{X}}$ can be obtained by the following formula:

$$\mathscr{C}_{\mathscr{X}} = \underset{\mathscr{C}_{\mathscr{X}}}{arg\min} \sum_{i=1}^{t} \sum_{j=1}^{N} \left\| \boldsymbol{x}_i - \boldsymbol{c}_j \right\|_2^2. \tag{10}$$

Let $d_{\max}$ denote the maximum distance between all samples in $\mathscr{X}$, σ can be estimated by the following formula:

$$\hat{\sigma} = \frac{d_{\max}}{\sqrt{2t}}. \tag{11}$$

Hence, the input layer of an ESN can be redefined as:

$$\mathscr{I}\left(\boldsymbol{x}_\tau; \mathscr{C}_{\mathscr{X}}, \hat{\sigma}\right) = \begin{bmatrix} \mathscr{K}\left(\boldsymbol{x}_\tau, \boldsymbol{c}_1; \hat{\sigma}\right) \\ \mathscr{K}\left(\boldsymbol{x}_\tau, \boldsymbol{c}_2; \hat{\sigma}\right) \\ \vdots \\ \mathscr{K}\left(\boldsymbol{x}_\tau, \boldsymbol{c}_N; \hat{\sigma}\right) \end{bmatrix}. \tag{12}$$

Hence, it follows that Eq.(1) can be rewritten as:

$$\tilde{\boldsymbol{s}}_\tau = \tanh\left(\mathscr{I}\left(\boldsymbol{x}_\tau; \mathscr{C}_{\mathscr{X}}, \hat{\sigma}\right) + \mathrm{V}\boldsymbol{s}_{\tau-1}\right). \tag{13}$$

LEMMA 1 (Echo state property [29]). *We let $\mathscr{X}^{-\tau} = \{\boldsymbol{x}_{t-\tau}, ..., \boldsymbol{x}_t\}$ denote an ordered set of finite inputs. Let $\mathscr{X}^{-\infty}$ denote a left infinite ordered input set $\mathscr{X}^{-\infty} = \{..., \boldsymbol{x}_{t-\tau}, ..., \boldsymbol{x}_{t-1}, \boldsymbol{x}_t\}$, and $\boldsymbol{s}_\tau = \mathscr{T}\left(\boldsymbol{x}_\tau, \boldsymbol{s}_{\tau-1}\right)$. $\forall \boldsymbol{x}_\tau \in \mathscr{X}^{-\infty}$, if there is another $\boldsymbol{s}_\tau^{'} = \mathscr{T}(\boldsymbol{x}_\tau, \boldsymbol{s}_{\tau-1}^{'})$, then $\boldsymbol{s}_\tau = \boldsymbol{s}_\tau^{'}(\tau \to t)$.*

Definition 1 (KESN has the echo state property). *When $\rho(\mathrm{V}) < 1$, KESN has the echo state property.*

Proof. Let us consider:

$$\begin{aligned} \left\| \boldsymbol{s}_\tau - \boldsymbol{s}'_\tau \right\| &= \left\| \mathscr{T}\left(\boldsymbol{x}_\tau, \boldsymbol{s}_{\tau-1}\right) - \mathscr{T}\left(\boldsymbol{x}_\tau, \boldsymbol{s}'_{\tau-1}\right) \right\| \\ &= \left\| \alpha \tanh\left(\mathscr{I}\left(\boldsymbol{x}_\tau\right) + \mathrm{V}\boldsymbol{s}_{\tau-1}\right) - \alpha \tanh\left(\mathscr{I}\left(\boldsymbol{x}_\tau\right) + \mathrm{V}\boldsymbol{s}'_{\tau-1}\right) + (1-\alpha)\boldsymbol{s}_{\tau-1} - (1-\alpha)\boldsymbol{s}'_{\tau-1} \right\| \\ &\leqslant \left\| \alpha \tanh\left(\mathscr{I}\left(\boldsymbol{x}_\tau\right) + \mathrm{V}\boldsymbol{s}_{\tau-1}\right) - \alpha \tanh\left(\mathscr{I}\left(\boldsymbol{x}_\tau\right) + \mathrm{V}\boldsymbol{s}'_{\tau-1}\right) \right\| + \left\| (1-\alpha)\boldsymbol{s}_{\tau-1} - (1-\alpha)\boldsymbol{s}'_{\tau-1} \right\| \\ &\leqslant \alpha \left\| \mathrm{V}\boldsymbol{s}_{\tau-1} - \mathrm{V}\boldsymbol{s}'_{\tau-1} \right\| + (1-\alpha) \left\| \boldsymbol{s}_{\tau-1} - \boldsymbol{s}'_{\tau-1} \right\| \\ &= \alpha\rho(\mathrm{V}) \left\| \boldsymbol{s}_{\tau-1} - \boldsymbol{s}'_{\tau-1} \right\| + (1-\alpha) \left\| \boldsymbol{s}_{\tau-1} - \boldsymbol{s}'_{\tau-1} \right\| \\ &= \left(\alpha\rho(\mathrm{V}) + (1-\alpha)\right) \left\| \boldsymbol{s}_{\tau-1} - \boldsymbol{s}'_{\tau-1} \right\| \end{aligned}. \tag{14}$$

This obviously satisfies the Lipschitz condition. Let $\alpha\rho(\mathrm{V}) + (1-\alpha) = \Lambda$, $\Lambda < 1$, then $\rho(\mathrm{V}) < 1$. So the distance between the two states shrinks by a scaling factor Λ. KESN satisfies the echo state property (Lemma 1).

□

3.2. Solver: kernel ridge regression

Let $\mathrm{W} = [w_1, w_2, ..., w_{N+K}]^{\mathrm{T}}$, $\mathscr{Y} = [y_1, y_2, ..., y_t]^{\mathrm{T}}$, $\boldsymbol{z}_\tau = \begin{bmatrix} \boldsymbol{x}_\tau \\ \boldsymbol{s}_\tau \end{bmatrix}$. $\phi : \mathbb{R}^p \to \mathbb{R}^q$ is a mapping. Hence, for kernel ridge regression, it follows that Eq.(8) can be rewritten as:

$$\hat{\mathrm{W}} = \underset{\mathrm{W}}{arg\min} \frac{1}{2} \sum_{\tau=1}^{t} \left(\mathrm{W}^{\mathrm{T}} \phi\left(\boldsymbol{z}_\tau\right) - y_\tau\right)^2 + \frac{\lambda}{2} \left\| \mathrm{W} \right\|_2^2. \tag{15}$$

Since ϕ is usually unknown, in order to get $\hat{\mathrm{W}}$, we make:

$$\mathscr{L} = \frac{1}{2}\sum_{\tau=1}^{t}\left(\mathrm{W}^{\mathrm{T}}\phi\left(z_\tau\right)-y_\tau\right)^2+\frac{\lambda}{2}\left\|\mathrm{W}\right\|_2^2. \tag{16}$$

We denote an $n\times n$ identity matrix by I_n. Let $\frac{\partial\mathscr{L}}{\partial\mathrm{W}}=0$, we can get:

$$\sum_{\tau=1}^{t}\left(\mathrm{W}^{\mathrm{T}}\phi\left(z_\tau\right)-y_\tau\right)\phi\left(z_\tau\right)+\lambda\mathrm{W}=0, \tag{17}$$

$$\hat{\mathrm{W}}=\left(\lambda\mathrm{I}_{(N+K)}+\sum_{\tau=1}^{t}\phi\left(z_\tau\right)\phi\left(z_\tau\right)^{\mathrm{T}}\right)^{-1}\left(\sum_{j=1}^{t}y_\tau\phi\left(z_j\right)\right). \tag{18}$$

Let:

$$\Phi=\left[\phi\left(z_1\right),\phi\left(z_2\right),...,\phi\left(z_\tau\right)\right]=\phi\left(\mathrm{S}\right). \tag{19}$$

Hence, we have:

$$\hat{\mathrm{W}}=\left(\lambda\mathrm{I}_{(N+K)}+\Phi\Phi^{\mathrm{T}}\right)^{-1}\Phi\mathscr{Y}. \tag{20}$$

The prediction function is:

$$\mathscr{F}\left(z\right)=\hat{\mathrm{W}}^{\mathrm{T}}\phi\left(z\right). \tag{21}$$

LEMMA 2 (Matrix inversion lemma [16]). *Let the block matrix of matrix* M *is:* $\mathrm{M}=\begin{bmatrix}\mathrm{A} & \mathrm{B}\\ \mathrm{C} & \mathrm{D}\end{bmatrix}$, *where* A *and* D *are invertible matrices. The following identity holds:*

$$\left(\mathrm{A}-\mathrm{BD}^{-1}\mathrm{C}\right)^{-1}\mathrm{BD}^{-1}=\mathrm{A}^{-1}\mathrm{B}\left(\mathrm{D}-\mathrm{CA}^{-1}\mathrm{B}\right)^{-1}. \tag{22}$$

Let $\mathrm{A}=\mathrm{I}_{(N+K)}$, $\mathrm{B}=\Phi$, $\mathrm{C}=-\Phi^{T}$, $\mathrm{D}^{-1}=\lambda^{-1}\mathrm{I}_t$, we have:

$$\left(\mathrm{I}_{(N+K)}+\lambda^{-1}\Phi\Phi^{\mathrm{T}}\right)^{-1}\lambda\Phi=\Phi\left(\lambda\mathrm{I}_t+\Phi^{\mathrm{T}}\Phi\right)^{-1}. \tag{23}$$

Hence, $\hat{\mathrm{W}}=\Phi\left(\lambda\mathrm{I}_t+\Phi^{\mathrm{T}}\Phi\right)^{-1}\mathscr{Y}$, and $\mathscr{F}\left(z\right)=\mathscr{Y}^{\mathrm{T}}\left(\lambda\mathrm{I}_t+\Phi^{\mathrm{T}}\Phi\right)^{-1}\Phi^{\mathrm{T}}\phi\left(z\right)$. Let $\mathrm{K}\in\mathbb{R}^{t\times t}$, $\mathrm{K}_{ij}=\phi\left(z_i\right)^{\mathrm{T}}\phi\left(z_j\right)=\mathscr{K}\left(z_i,z_j\right)$, we have:

$$\mathscr{F}\left(z\right)=\mathscr{Y}^{\mathrm{T}}\left(\lambda\mathrm{I}_t+\mathrm{K}\right)^{-1}\mathscr{K}\left(z,\mathrm{S}\right), \tag{24}$$

where $\mathscr{K}$ is a kernel function such as Eq.(9).

3.3. Optimizing hyper-parameters: tree-structured parzen estimator approach

Tree-structured Parzen Estimator Approach(TPE) is a sequential model-based global optimization(SMBO) model [27], which is also a Bayesian optimization method. Many times directly evaluating our model is costly, so SMBO uses a surrogate model to evaluate parameter combinations in the hyper-parameter space to find an optimal parameter to evaluate our model. TPE is efficient and performs better than many other SMBO models. In addition, TPE can search in high-dimensional hyperparameter spaces. More importantly, TPE can set different probability density distributions such as log-uniform and normal distribution for each parameter. In addition, TPE also supports optimization directly from the given parameter space.

In this paper, we use the results of 1000 iterations of TPE as the optimal hyper-parameter combination. The specific hyper-parameter search space will be given in Section.4. The complete training and prediction process of DKESN is presented in Algorithm 2.

Algorithm 2: DKESN

Input:
Train set: $\mathscr{X}_{\text{train}} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_t]^{\mathrm{T}}, \mathscr{Y}_{\text{rain}} = [\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_t]^{\mathrm{T}}$;
Test set: $\mathscr{X}_{\text{test}} = [\boldsymbol{x}_{t+1}, \boldsymbol{x}_{t+2}, ..., \boldsymbol{x}_{t+k}]^{\mathrm{T}}$;
reservoir size: N;
leak rate: α;
spectral radius: ρ_r ;
penalty coefficient: λ;
Output: $\hat{\boldsymbol{y}} = \text{DKESN}(\boldsymbol{x})$

$\mathrm{V} \leftarrow$ random matrix $\in \mathbb{R}^{N \times N}$;
$\mathrm{V} \leftarrow \frac{\rho_r}{\rho(\mathrm{V})}\mathrm{V}$; // Eq.(6)
$\boldsymbol{s}_0 \leftarrow 0$;
$\mathscr{C}_{\mathscr{X}} \leftarrow \underset{\mathscr{C}_{\mathscr{X}}}{arg\min} \sum_{i=1}^{t}\sum_{j=1}^{N} \left\|\boldsymbol{x}_i - \boldsymbol{c}_j\right\|_2^2 (j = 1,2,...,N)$; // Eq.(10)
for $\tau = 1$ *to* t **do**
  $\tilde{\boldsymbol{s}}_\tau \leftarrow \tanh\left(\mathscr{I}\left(\boldsymbol{x}_\tau; \mathscr{C}_{\mathscr{X}}, \sigma\right) + \mathrm{V}\boldsymbol{s}_{\tau-1}\right)$; // Eq.(13)
  $\boldsymbol{s}_\tau \leftarrow \alpha\tilde{\boldsymbol{s}}_\tau + (1-\alpha)\boldsymbol{s}_{\tau-1}$; // Eq.(2)
end
$\mathrm{S} \leftarrow \begin{bmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_t \\ \boldsymbol{s}_1 & \boldsymbol{s}_1 & & \boldsymbol{s}_t \end{bmatrix}$; // Eq.(4)
for $\tau = t+1$ *to* $t+k$ **do**
  $\tilde{\boldsymbol{s}}_\tau \leftarrow \tanh\left(\mathscr{I}\left(\boldsymbol{x}_\tau; \mathscr{C}_{\mathscr{X}}, \sigma\right) + \mathrm{V}\boldsymbol{s}_{\tau-1}\right)$; // Eq.(13)
  $\boldsymbol{s}_\tau \leftarrow \alpha\tilde{\boldsymbol{s}}_\tau + (1-\alpha)\boldsymbol{s}_{\tau-1}$; // Eq.(2)
  $\hat{y}_{\tau-t} = \mathscr{F}\left(\begin{bmatrix} \boldsymbol{x}_\tau \\ \boldsymbol{s}_\tau \end{bmatrix}\right)$; // Eq.(24)
end

4. SIMULATION RESULTS

In this section, we use chaotic time series generated by the Mackey-Glass system for the numerical simulation of DKESN. Furthermore, this system is the most used chaotic systems for validating ESN models [6,9,26]. All data and computer programs are available from github: `https://github.com/tang-guoxin/dkesn`. In addition, we compare DKESN with RESN, LESN, $\ell_{1/2}$-ESN, and EESN. the structure and training algorithm of DKESN are presented in Figure 1(b) and Algorithm 2, and the structure and training algorithm of the remaining ESN models are given in Figure 1(a) and Table 1. the training algorithm of RESN has also been given in Algorithm 1.

The mathematical model of the Mackey-Glass system is:

$$\frac{dx}{dt} = \beta_1 x(t) + \frac{\beta_2 x(t-\tau)}{1 + x^q(t-\tau)}. \tag{25}$$

When $\tau > 16.8$, the system has chaotic attractor. This paper takes $\tau = 17$, $q = 10$, $\beta_1 = -0.1$, and $\beta_2 = 0.2$. The images of the first 500 moments of the system are shown in Figure 2.

Let the hyper-parameters set is $\mathscr{H} = \{N, \alpha, \rho_r, \lambda, \mu\}$. Among them, $\{N, \alpha, \rho_r\}$ are the inherent hyper-parameters of ESN, λ is the ℓ_1 regularization parameter, μ is the elastic-net regularization parameter ($\lambda\left(\mu\ell_1 + (1-\mu)\ell_2\right)$). The evaluation metrics of the model uses the mean square error (MSE):

$$\text{MSE} = \frac{1}{t}\sum_{\tau}^{t}(y_\tau - \hat{y}_\tau)^2. \tag{26}$$

The search space and optimal hyper-parameters of hyper-parameters are shown in Table 2:

The optimal parameters given by TPE in Table 2 were used as parameters for each model. We use 2000 data as training set and 1000 data as test set. Figure 3 shows the predicted performance of the last 300 data. It can be found that the performance of DKESN is the best, the performance of RESN is also generally satisfactory, while the performance of LESN, EESN and $\ell_{1/2}$-ESN is poor.

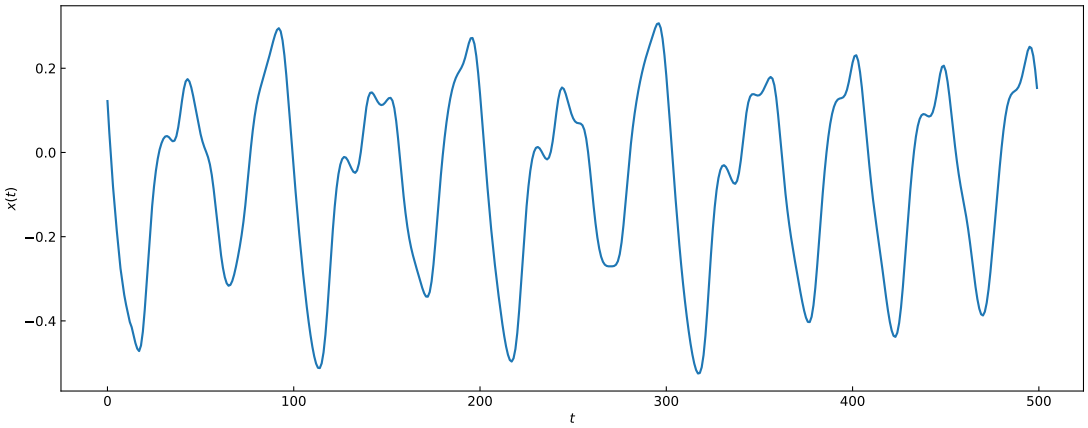

Fig. 2 – Mackey-Glass system.

Table 2

The hyper-parameters space and the optimal hyper-parameters

$\mathscr{H}$	Space	Number of parameters	RESN	LESN	$\ell_{1/2}$-ESN	EESN	DKESN
N	$[100,1000]$	10	900	700	700	800	800
α	$[0.2,0.5]$	30	0.26	0.29	0.29	0.30	0.30
ρ_r	$[0.8,1.5]$	50	1.48	0.83	0.97	1.14	1.25
λ	$[10^{-1},10^{-9}]$	9	10^{-3}	10^{-4}	10^{-8}	10^{-7}	10^{-8}
μ	$[0.25,0.75]$	10	-	-	-	0.47	-

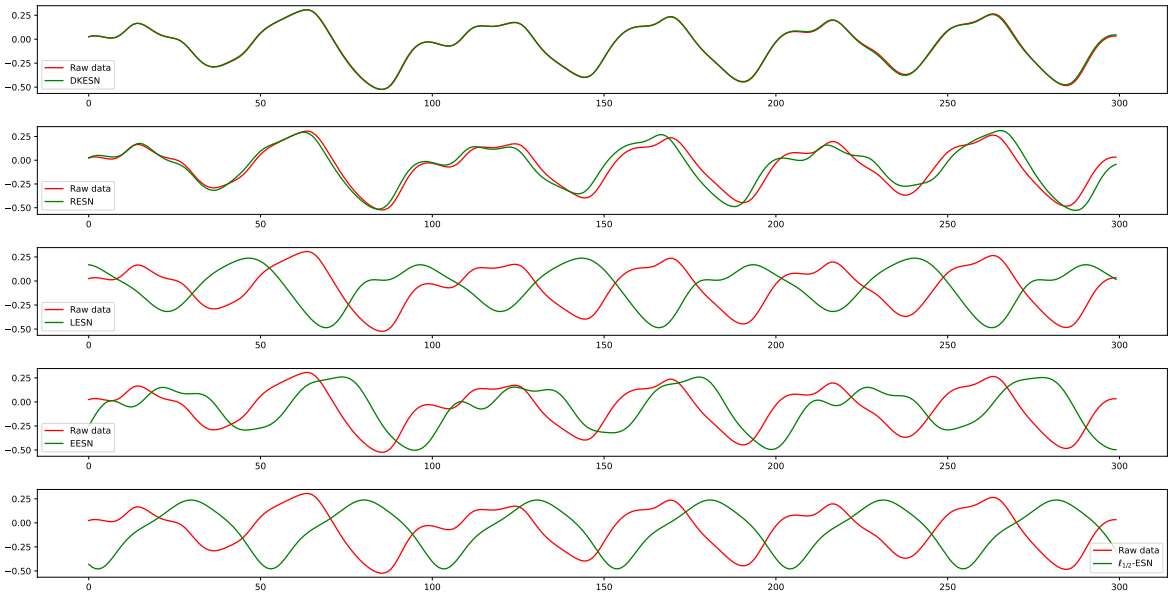

Fig. 3 – Raw data and predicted values.

In order to check the stability of the model, we use 3000 data as the training set of the five models, and take the data sizes of 500, 1000, 1500, 2000, 2500 and 3000 as the test set respectively. In addition, each model was retrained 20 times, and the test set error of the model was recorded, and the 95% confidence interval of each model error was calculated separately.

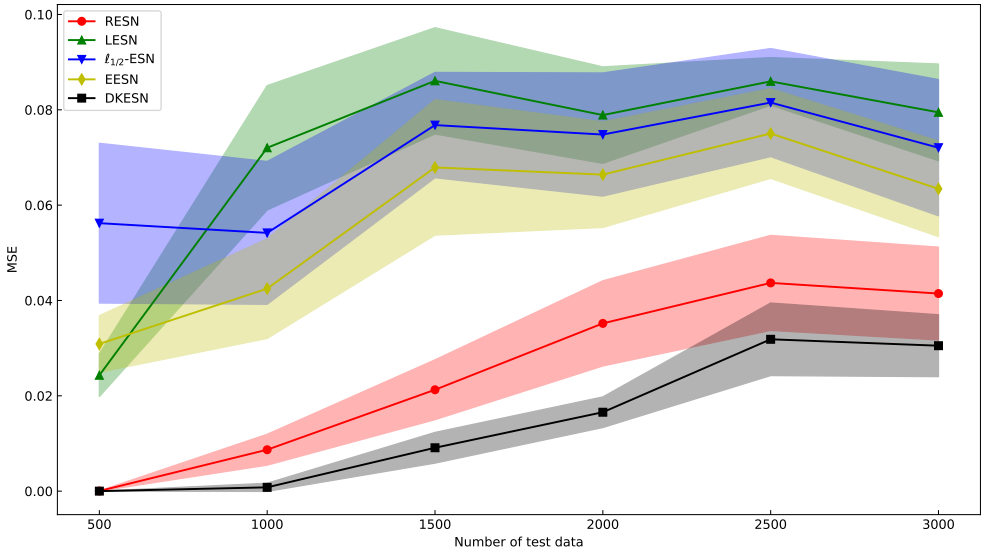

Fig. 4 – Test error (MSE) and 95% confidence interval.

Figure 4 shows the performance of each model intuitively. It can be found that the performance of LESN, EESN, and $\ell_{1/2}$-ESN is poor and the range of confidence interval is large. Confidence intervals for RESN and DKESN have a narrower range. Overall, DKESN always has the smallest confidence interval, and MSE is also the smallest of all models. This shows that the model in this paper has better performance. In addition, from the sparsity of the model, $\ell_{1/2} - \text{ESN} \prec \text{LESN} \prec \text{EESN} \prec \text{RESN} \prec \text{DKESN}$. However, since the reservoir layer of ESN is a sparse matrix, we can think that each output of the reservoir layer contains very little information. If we continue to perform feature compression or feature selection on the model, this will continue to cause loss of information and ultimately lead to performance degradation.

From another point of view, the reservoir layer is equivalent to a dropout operation, which makes some weights not work: the weight is 0. In deep neural networks (DNNs) we often use dropout to reduce the risk of overfitting. The reservoir layer, as a sparse matrix, has many elements equal to 0. And the exciting thing is that for DNN, dropout only works when training the model, and when the model will be validated with the test set, dropout does not work. However, the reservoir matrix is a fixed sparse matrix, which means that we perform dropout operations for both training and testing. Hence, it is not necessary to continue to compress and select model features at the output layer, as this may lead to underfitting.

From the perspective of the size of the confidence interval, using the kernel function as the mapping of the input layer is more stable than the model obtained by directly generating a random matrix U as the weight of the input layer. When an inappropriate input layer matrix U is generated, for example, the matrix has strong collinearity, which may lead to the inability to effectively extract the input features, and eventually lead to the direct collapse of the entire model. Using the kernel function completely avoids this problem because the output of the input layer depends only on the data itself.

5. CONCLUSION

In this work, we propose a kernel echo state network trained using kernel ridge regression: DKESN. We prove that DKESN has the echo state property. DKESN removes the random weight matrix of the input layer of ESN. We use the k-means algorithm and kernel function to map the input to a high-dimensional space, which is controllable. In other words, the result of the input layer depends entirely on the reservoir size N and the original data. In addition, in the existing studies, the way to obtain W is a simple linear model, which may perform poorly in data with strong nonlinearity. Hence, KRR is a more suitable choice. For the reservoir layer, we consider this to be a dropout method that alleviates overfitting and improves generalization. Experiments show that our proposed model has better performance and higher stability.

Since the input and reservoir layer matrices do not have to be trained, an "excellent" reservoir matrix is helpful to improve the performance of DKSEN. In the future, how to determine a reservoir matrix is a value of in-depth research.

ACKNOWLEDGEMENTS

This work is supported by grants from the National Natural Science Foundation of China (No. 12075162), the Sichuan Federation of Social Science Associations (No. SC20TJ016), the Sichuan Science and Technology Department (No. 2020YJ0357) and the VC & VR Key Laboratory of Sichuan Province, the Graduate Research and Innovation Foundation of Chongqing, China (CYS22074).

CONFLICT OF INTEREST

The authors declare no conflicts of interest.

DATA AVAILABILITY STATEMENT

All data and computer programs are available from github: `https://github.com/tang-guoxin/dkesn`.

REFERENCES

1. L. YU, X. MA, W. WU, X. XIANG, Y. WANG, B. ZENG, *Application of a novel time-delayed power-driven grey model to forecast photovoltaic power generation in the Asia-Pacific region*, Sustainable Energy Technologies and Assessments, **44**, art. 100968, 2021.
2. L. YU, X. MA, W. WU, Y. WANG, B. ZENG, *A novel elastic net-based NGBMC (1,n) model with multi-objective optimization for nonlinear time series forecasting*, Communications in Nonlinear Science and Numerical Simulation, **96**, art. 105696, 2021.
3. C. CORTES, V. VAPNIK, *Support-vector networks*, Machine learning, **20**, *3*, pp. 273–297, 1995.
4. J.F. QIAO, H.G. HAN, *Identification and modeling of nonlinear dynamical systems using a novel self-organizing RBF-based approach*, Automatica, **48**, *8*, pp. 1729–1734, 2012.
5. I. GOODFELLOW, Y. BENGIO, A. COURVILLE, *Deep learning*, MIT press, 2016.
6. H. JAEGER, H. HAAS, *Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication*, Science, **304**, *5667*, pp. 78–80, 2004.
7. A.H. RIBEIRO, K. TIELS, L.A. AGUIRRE, T/ SCHÖN, *Beyond exploding and vanishing gradients: analysing RNN training using attractors and smoothness*, Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, 2020, PMLR, vol. 108, pp. 2370–2380.
8. M. LUKOŠEVIČIUS, *A practical guide to applying echo state networks*, in: *Neural networks: Tricks of the trade*, Springer, 2012, pp. 659–686.
9. J. QIAO, L. WANG, C. YANG, *Adaptive lasso echo state network based on modified Bayesian information criterion for nonlinear system modeling*, Neural Computing and Applications, **31**, *10*, pp. 6163–6177, 2019.
10. M. XU, M. HAN, *Adaptive elastic echo state network for multivariate time series prediction*, IEEE transactions on Cybernetics, **46**, *10*, pp. 2173–2183, 2016.
11. L. WANG, Z. SU, J. QIAO, C. YANG, *Design of sparse Bayesian echo state network for time series prediction*, Neural Computing and Applications, **33**, pp. 7089–7102, 2021.
12. C. YANG, J. QIAO, Z. AHMAD, K. NIE, L. WANG, *Online sequential echo state network with sparse RLS algorithm for time series prediction*, Neural Networks, **118**, pp. 32–42, 2019.
13. X. YAO, Z. WANG, *Fractional order echo state network for time series prediction*, Neural Processing Letters, **52**, *1*, pp. 603–614, 2020.
14. H. HU, L. WANG, S.X. LV, *Forecasting energy consumption and wind power generation using deep echo state network*, Renewable Energy, **154**, pp. 598–613, 2020.
15. P.L. MCDERMOTT, C.K. WIKLE, *An ensemble quadratic echo state network for non-linear spatio-temporal forecasting*, Stat, **6**, *1*, pp. 315–330, 2017.

16. C. ROBERT, *Machine learning, a probabilistic perspective*, Taylor & Francis, 2014.
17. C. GALLICCHIO, A. MICHELI, L. PEDRELLI, *Deep reservoir computing: A critical experimental analysis*, Neurocomputing, **268**, pp. 87–99, 2017.
18. Q. MA, L. SHEN, G.W. COTTRELL, *DeePr-ESN: A deep projection-encoding echo-state network*, Information Sciences, **511**, pp. 152–171, 2020.
19. J. FRIEDMAN, T. HASTIE, R. TIBSHIRANI, *Regularization paths for generalized linear models via coordinate descent*, Journal of Statistical Software, **33**, *1*, pp. 1–22, 2010.
20. B. EFRON, T. HASTIE, I. JOHNSTONE, R. TIBSHIRANI, *Least angle regression*, The Annals of Statistics, **32**, *2*, pp. 407–499, 2004.
21. E.M. RONCHETTI, P.J. HUBER, *Robust statistics*, John Wiley & Sons, 2009.
22. C.M. BISHOP, N.M. NASRABADI, *Pattern recognition and machine learning*, vol. 4, Springer, 2006.
23. R. KOENKER, G. BASSETT JR, *Regression quantiles*, Econometrica: Journal of the Econometric Society, **46**, *1*, pp. 33–50, 1978.
24. P. BALDI, P.J. SADOWSKI, *Understanding dropout*, in: *Advances in Neural Information Processing Systems*, vol. 26, 2013.
25. Z. XU, H. ZHANG, Y. WANG, X. CHANG, Y. LIANG, $L_{1/2}$ *regularization*, Science China Information Sciences, **53**, *6*, pp. 1159–1169, 2010.
26. J. QIAO, F. LI, H. HAN, W. LI, *Growing echo-state network with multiple subreservoirs*, IEEE Transactions on Neural Networks and Learning Systems, **28**, *2*, pp. 391–404, 2016.
27. J. BERGSTRA, R. BARDENET, Y. BENGIO, B. KÉGL, *Algorithms for hyper-parameter optimization*, in: *Advances in Neural Information Processing Systems*, vol. 24, 2011.
28. D.S. BROOMHEAD, D. LOWE, *Radial basis functions, multi-variable functional interpolation and adaptive networks*, Technical report, Royal Signals and Radar Establishment Malvern (United Kingdom), 1988.
29. H. JAEGER, HERBERT, *The "echo state" approach to analysing and training recurrent neural networks – with an erratum note* (corrected version 2010), GMD Report 148, German National Research Center for Information Technology, Bonn, Germany, 2001.

Received October 13, 2022