Contents lists available at ScienceDirect

Applied Soft Computing

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

Multi-Reservoir Echo State Networks with Hodrick–Prescott Filter for nonlinear time-series prediction[☆,☆☆]

Ziqiang Li [a,*], Yun Liu [b], Gouhei Tanaka [a,c,*]

[a] Department of Electrical Engineering and Information Systems, Graduate School of Engineering, The University of Tokyo, Hongo, Bunkyo-ku, 113-8656, Tokyo, Japan
[b] Department of Computer Science, School of Computing, Tokyo Institute of Technology, Ookayama, Meguro-ku, 152-8550, Tokyo, Japan
[c] International Research Center for Neurointelligence, The University of Tokyo, Hongo, Bunkyo-ku, 113-0033, Tokyo, Japan

A R T I C L E I N F O

Article history:
Received 8 November 2021
Received in revised form 29 December 2022
Accepted 6 January 2023
Available online 18 January 2023

Keywords:
Neural networks
Reservoir computing
Time series decomposition
Time series prediction

A B S T R A C T

The Echo State Network (ESN) is a representative model for reservoir computing, which is capable of high-speed model training for machine learning tasks with time series data. Extended models of the ESN, such as Multi-Reservoir ESNs (MRESNs), have been intensively studied for performance improvement in recent years. In this study, we propose a new model called an HP-MRESN by combining an MRESN with the Hodrick–Prescott (HP) filter for nonlinear time series prediction. The proposed HP-MRESN comprises three basic components: a time series decomposer, a reservoir state extractor, and an ensemble decoder. In the time series decomposer, we recursively leverage the HP filter to decompose original time-series data into multiple trend and cycle components. In the reservoir state extractor, each time series component is fed into a corresponding reservoir-state encoder for generating a reservoir state which is extracted as it is or through the principal component analysis. In the ensemble decoder, the states of multiple reservoirs are collected and processed to produce model outputs. Moreover, we propose a greedy algorithm to automatically find the best model architectures under designated hyperparameters for different prediction tasks. Experimental results on a total of 24 nonlinear time-series prediction tasks with 6 real-world datasets demonstrate that our proposed HP-MRESN not only can outperform some existing representative MRESN models and fully-trained RNN models but also can have relatively low training time. In addition, performance comparisons between the HP-MRESN and related MRESN models with other prepossessing methods show the benefit of time series decompositions using the HP filter. The codes of the proposed method are publicly available on https://github.com/Ziqiang-IRCN/HP-MRESN.

1. Introduction

Nonlinear Time-series Prediction (NTP) [1] is one of the popular research topics in machine learning due to broad demands in different regions [2]. The main purpose of NTP tasks is to predict a future state of a nonlinear dynamic system based on previously observed data. Since Neural Networks (NNs) can have a powerful ability of approximating a dynamic system behind the observed time series data, many NN-based methods have been successfully applied to NTP tasks [3]. In particular, Recurrent Neural Networks (RNNs) [4], as an important branch of NNs, have been widely used for NTP tasks. Benefiting from the outstanding temporal dependency representation ability brought about by recurrent feedback connections, RNNs outperform other classical NN-based methods on NTP tasks [3]. However, the recurrent architecture of RNN frequently induces gradient explosion and/or vanishing in the training process [5]. Even though some variants of RNN such as Long Short-Term Memory (LSTM) [6] and Gated Recurrent Unit (GRU) [7] can significantly relieve the gradient explosion and vanishing problems, they still cannot avoid these problems perfectly. On the other hand, the Backpropagation Through Time (BPTT) [8] normally used for training the RNN variants often suffers from high computational cost and local optima problem [9]. These problems are a bottleneck in obtaining a well-trained RNN model.

Reservoir Computing (RC) [11,12] is an alternative computational framework that provides a remarkably efficient approach for training RNN. The most important characteristic of RC is that

[☆] The affiliation of Ziqiang Li has been changed in the period of revision as follows: International Research Center for Neurointelligence, The University of Tokyo, Hongo, Bunkyo-ku, 113-0033, Tokyo,Japan.
[☆☆] The affiliation of Yun Liu has been changed in the period of revision as follows: Artificial Intelligence Research Center, National Institute of Advanced Industrial Science and Technology (AIST), Aomi, Koto-ku, 135-0064, Tokyo, Japan.
[*] Corresponding authors.
E-mail addresses: ziqiang-li@g.ecc.u-tokyo.ac.jp (Z. Li), liu.yun@aist.go.jp (Y. Liu), gtanaka@g.ecc.u-tokyo.ac.jp (G. Tanaka).

https://doi.org/10.1016/j.asoc.2023.110021

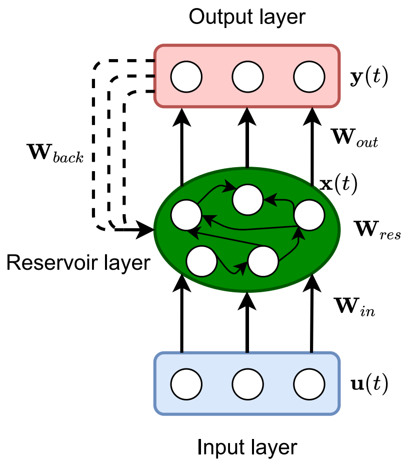

Fig. 1. A schematic diagram of a standard ESN [10].

a predetermined non-linear system (i.e. reservoir) is used to map input time series data into a high-dimensional feature space and then a readout mechanism with a simple learning algorithm is used to generate output time series from the high-dimensional reservoir state. This characteristic enables training RNNs with extremely low computational cost. The Echo State Network (ESN) is a popular form of an RC model. We show a schematic diagram of a standard ESN in Fig. 1 [10]. A standard ESN is composed of three layers: an input layer, a reservoir layer, and an output layer. The neurons of the input layer and those of the reservoir layer are fully connected with fixed weights represented as an input weight matrix $\mathbf{W}_{in}$. The neurons inside the reservoir layer are connected with fixed recurrent weights represented as a predefined sparse weight matrix $\mathbf{W}_{res}$. The trainable output weight matrix, $\mathbf{W}_{out}$, is set to map the reservoir state to a desired output by a simple linear regression method. The feedback weight matrix indicated by dashed lines, $\mathbf{W}_{back}$, represents optional connection weights from the output layer to the reservoir layer. Many studies have reported that ESN-based methods can outperform fully-trained RNNs for some NTP benchmark datasets [13] with much less training cost. However, the relatively fixed training scheme and the simple architecture of the standard ESN unavoidably limit its representation ability and prediction performances on NTP tasks.

In order to solve this dilemma, many ESN-based models with multiple reservoirs were proposed recently [14,15,17]. We show the schematic diagram of three representative models in Fig. 2. Similarly to the stack architecture of deep NNs, Deep Echo State Network (DeepESN) [14] is designed by stacking multiple reservoir layers with inter-reservoir connections as shown in Fig. 2(a). Several studies pointed out that the structured reservoir state spaces can significantly improve prediction performances on some NTP tasks [18,19]. Other models are constructed by hierarchically connecting multiple standard ESNs [15,16,20], which we call Multi-Step Learning ESN (MSLESN) hereafter, as shown in Fig. 2(b). Their difference from the DeepESN is that the learning of the output weight matrices in the multiple ESNs are performed not in one shot but one by one. Related studies reported that this kind of model performs well not only in NTP tasks but also in image recognition tasks [21]. Another representative model called Grouped Echo State Network (GroupedESN) [17] follows a tiled layout where multiple independent reservoirs are placed in parallel to share the same input data as shown in Fig. 2(c). With the benefits of extended reservoir state spaces, their representation ability can be improved and the corresponding prediction performance can be much better than that of

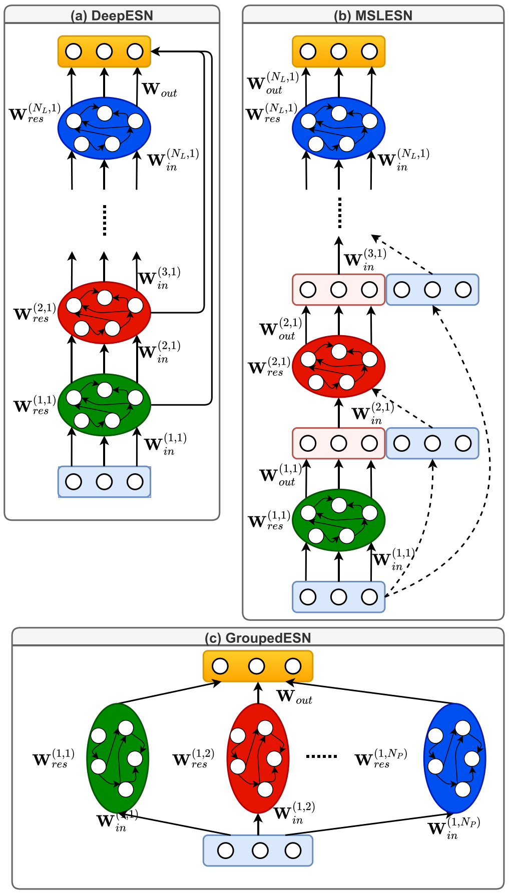
(a) DeepESN
$\mathbf{W}_{out}$
$\mathbf{W}_{res}^{(N_L,1)}$
$\mathbf{W}_{in}^{(N_L,1)}$
$\mathbf{W}_{in}^{(3,1)}$
$\mathbf{W}_{res}^{(2,1)}$
$\mathbf{W}_{in}^{(2,1)}$
$\mathbf{W}_{res}^{(1,1)}$
$\mathbf{W}_{in}^{(1,1)}$
(b) MSLESN
$\mathbf{W}_{out}^{(N_L,1)}$
$\mathbf{W}_{res}^{(N_L,1)}$
$\mathbf{W}_{in}^{(N_L,1)}$
$\mathbf{W}_{in}^{(3,1)}$
$\mathbf{W}_{out}^{(2,1)}$
$\mathbf{W}_{res}^{(2,1)}$
$\mathbf{W}_{in}^{(2,1)}$
$\mathbf{W}_{out}^{(1,1)}$
$\mathbf{W}_{res}^{(1,1)}$
$\mathbf{W}_{in}^{(1,1)}$
(c) GroupedESN
$\mathbf{W}_{out}$
$\mathbf{W}_{res}^{(1,1)}$
$\mathbf{W}_{res}^{(1,2)}$
$\mathbf{W}_{res}^{(1,N_P)}$
$\mathbf{W}_{in}^{(1,1)}$
$\mathbf{W}_{in}^{(1,2)}$
$\mathbf{W}_{in}^{(1,N_P)}$

Fig. 2. Architectures of three representative MRESN-based models. (a) A DeepESN with N_L stacked reservoir layers [14]. (b) An MSLESN with N_L stacked ESNs. This architecture shows the common characteristic that the temporary outputs generated from the l-th ESN are fed into the $(l+1)$ ESN as inputs [15,16], and the dash lines indicate the optional connections. (c) A GroupedESN with N_P parallel reservoirs [17].

the standard ESN. Throughout this paper, we refer to these ESN models with multiple reservoirs as Multi-Reservoir Echo State Networks (MRESNs).

Based on the above-mentioned three representative models of MRESN, several successful variations for NTP tasks have been proposed [22–25]. One popular research direction of these variations is to add a prepossessing method for extracting significant features from the original input time series data. We notice that these variations are commonly based on the scheme of "decomposition-(parallel)encoding–decoding" in signal processing. For example, Discrete Wavelet Transform (DWT) [26] is used to decompose the original input before processing in multiple parallel ESNs. By multi-level decomposition and reconstruction, each reconstructed low-pass or high-pass signal is individually fed into one of the parallel ESNs as an input time series. The results reported in Ref. [22] demonstrate that the combination of wavelet transforms methods and parallel reservoirs can effectively enhance prediction performance. However, the convolution operations of DWT on a finite-length signal unavoidably cause distortion and corruption of reconstructed signals [27]. Moreover, it is burdensome to select a suitable basis function for DWT for a given specific time-series dataset. In Ref. [25], the Ensemble Empirical Mode Decomposition (EEMD) algorithm [28] is leveraged to decompose an original time series. To remove the white noise that is previously added to the original signal in the process of the EEMD, the Savitzky–Golay (SG) filter [29] is applied to the separated components at different resolutions. This extra smoothing process may induce information loss and extra computational cost. Some works proposed leveraging other machine learning methods such as Restricted Boltzmann Machines (RBMs) [23,24] to generate high-dimensional features from an original input time series instead of feeding raw time-series data into reservoirs. These methods can enhance the prediction performance by extracting rich features from the original input data. However, the corresponding computational cost of training RBMs is relatively high.

From the aforementioned works, we can see that most variants of the MRESN can outperform the prototype models whereas their decomposition methods face the issues of high computational cost and excessive hyperparameter settings. On the other hand, to our best knowledge, few works have focused on evaluating prediction performances of the MRESN-based models in NTP tasks with real-world time-series datasets, which hinders our understanding of whether the proposed MRESN-based model can satisfy the needs of real-world applications. Motivated by the above issues, this work aims to propose an MRESN-based hybrid model that can achieve higher prediction performance than some existing MRESN-based models in real data processing with relatively low computational cost. To accomplish this goal, we follow the basic scheme of "decomposition-(parallel)encoding–decoding" and put our focus on a time-series decomposition method with fewer calculations and parameters. The Hodrick–Prescott (HP) filter [30] is one of the classical time-series decomposition methods used originally in economics. The original aim of the HP filter was to distill long-term trends from economic time series data by a finely-designed formula with a smoothing factor. A recent study in economics pointed out that the HP filter may not be an ideal tool for extracting long-term trends in macroeconomics [31]. However, this drawback is not detrimental to the merit of high efficiency and simplicity brought about by its closed-form solution including only a single hyperparameter. In this paper, a novel MRESN-based model, HP-MRESN, is proposed by creatively combining HP filters together with multiple parallel reservoir state extractors. Typically, the HP filter decomposes a time-series data into a trend component and a cycle component. In our proposed method, we recursively decompose each cycle component by HP filters with different smoothing factors and feed each component time series into the corresponding reservoir state encoder for temporal feature extraction. The final model outputs are generated by summing partial outputs up together in the ensemble decoder.

The main contributions of this study are summarized as follows:

1. We propose the new model, HP-MRESN, for nonlinear time-series prediction tasks. Specifically, we show how to effectively use HP filters in decomposing the original input time series into multiple trend components and one cycle component.
2. We compare the prediction performance between the HP-MRESN and other representative MRESN models in various NTP tasks mainly with real-world time-series datasets. The experimental results show that the HP-MRESN outperforms the tested MRESNs and fully-trained RNNs in many of 24 tasks with 6 real-world benchmark NTP datasets.
3. We give a detailed theoretical comparison of the training cost between the proposed model and three typical MRESN models. The analytical results show that the HP-MRESN is comparable to the most efficient existing MRESN in terms of training cost under the same number of reservoirs. Moreover, we demonstrate in numerical experiments that the HP-MRESN is trained with much lower computational efforts than fully-trained RNNs (e.g., the ElmanNet [4], the LSTM, and the GRU) in the multi-output prediction task with real-world benchmark NTP datasets.

The rest of this paper is organized as follows: In Section 2, we give preliminaries of the modular architecture of MRESNs and describe the GroupedESN as a basis of our proposal. In Section 3, we introduce the details of the proposed HP-MRESN. In Section 4, we present a greedy algorithm for optimizing the architecture of the HP-MRESN and analyze the corresponding computational complexity. In Section 5, we show the details of numerical experiments. In Section 6, we give discussion about the experimental results and the significance of this work.

2. Preliminaries

In this paper, we use capital bold letters, lowercase bold letters, and lowercase letters to represent matrices, vectors, and scalers, respectively.

We introduce a modular architecture for constructing the MRESN models in Section 2.1. Then, we briefly describe the GroupedESN [17], which can be viewed as a basis of our proposed model, in Section 2.2.

In this study, we consider NTP tasks. Throughout this paper, we denote the original input time-series data and the corresponding target data to be predicted by matrices $\mathbf{U} = [\mathbf{u}(1), \mathbf{u}(2), \ldots, \mathbf{u}(N_T)]$ and $\mathbf{Y} = [\mathbf{y}(1), \mathbf{y}(2), \ldots, \mathbf{y}(N_T)]$, respectively, where the symbol N_T represents the time length. The dimensions of $\mathbf{u}(t)$ and $\mathbf{y}(t)$ are denoted by N_U and N_Y, respectively.

2.1. Modular architecture

By introducing a view of modular architecture, we can represent various MRESN models using specific layouts of encoders and decoders in a unified way. An encoder and a decoder are defined as follows [32]:

- An encoder Θ_{enc} is composed of an input weight matrix $\mathbf{W}_{in} \in \mathbb{R}^{N_R \times N_U}$ and a reservoir weight matrix $\mathbf{W}_{res} \in \mathbb{R}^{N_R \times N_R}$.
- A decoder Θ_{dec} is composed of an output weight matrix $\mathbf{W}_{out} \in \mathbb{R}^{N_Y \times N_R}$.

The function of the encoder is to map the input data into an N_R-dimensional reservoir state space. Each state in the reservoir space is transformed into a model output via the decoder so as to be close to the target data. According to Ref. [32], the arrangement of encoders is possible in two directions, i.e. the layered direction and the parallel direction. We add superscript (l, p) to Θ_{enc} and Θ_{dec} to indicate their specific positions in the two directions, which can be formulated as $\Theta_{enc}^{(l,p)} = \left\{\mathbf{W}_{in}^{(l,p)}, \mathbf{W}_{res}^{(l,p)}\right\}$ and $\Theta_{dec}^{(l,p)} = \left\{\mathbf{W}_{out}^{(l,p)}\right\}$ for $1 \leq l \leq N_L$ and $1 \leq p \leq N_P$. The symbols N_L and N_P are the maximal numbers of modules in the layered direction and the parallel direction, respectively.

2.2. GroupedESN

The GroupedESN owns N_P parallel encoders and one decoder as shown in Fig. 2(c). The corresponding reservoir states of $\Theta_{enc}^{(1,p)}$ can be calculated as follows:

$$\mathbf{x}^{(1,p)}(t) = (1-\alpha)\mathbf{x}^{(1,p)}(t-1) + \alpha \tanh\Big(\mathbf{W}_{in}^{(1,p)}\mathbf{s}^{(1,p)}(t) + \mathbf{W}_{res}^{(1,p)}\mathbf{x}^{(1,p)}(t-1)\Big), \quad \text{for } p = 1, 2, \ldots, N_P, \tag{1}$$

where α denotes the leaking rate [33] and $\mathbf{s}^{(1,p)}(t)$ is the input vector corresponding to $\Theta_{enc}^{(1,p)}$ at time t. The element values of $\mathbf{W}_{in}^{(1,p)}$ are randomly drawn from the uniform distribution in the range of $[-\delta, \delta]$, where δ represents the input scaling. The element values of $\mathbf{W}_{res}^{(1,p)}$ are randomly assigned from the uniform distribution in the range of $[-1, 1]$. By considering the Echo State

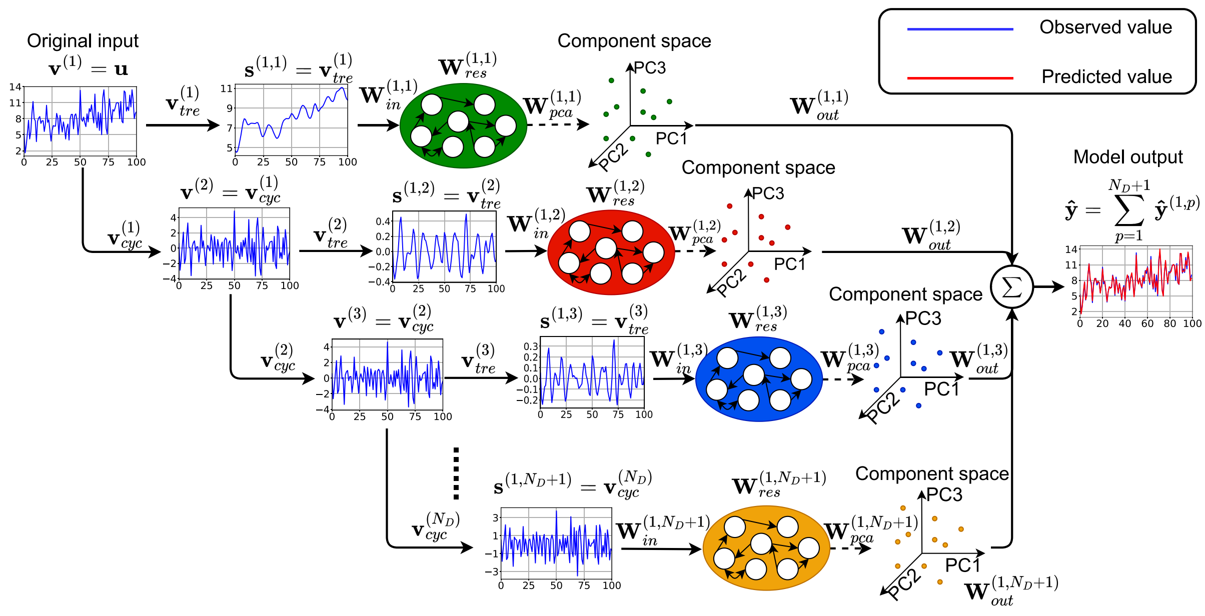
Original input
$\mathbf{v}^{(1)} = \mathbf{u}$
$\mathbf{v}_{tre}^{(1)}$
$\mathbf{s}^{(1,1)} = \mathbf{v}_{tre}^{(1)}$
$\mathbf{W}_{in}^{(1,1)}$
$\mathbf{W}_{res}^{(1,1)}$
$\mathbf{W}_{pca}^{(1,1)}$
Component space
PC1
PC2
PC3
$\mathbf{W}_{out}^{(1,1)}$
Observed value
Predicted value
$\mathbf{v}_{cyc}^{(1)}$
$\mathbf{v}^{(2)} = \mathbf{v}_{cyc}^{(1)}$
$\mathbf{v}_{tre}^{(2)}$
$\mathbf{s}^{(1,2)} = \mathbf{v}_{tre}^{(2)}$
$\mathbf{W}_{in}^{(1,2)}$
$\mathbf{W}_{res}^{(1,2)}$
$\mathbf{W}_{pca}^{(1,2)}$
$\mathbf{W}_{out}^{(1,2)}$
Model output
$\hat{\mathbf{y}} = \sum_{p=1}^{N_D+1} \hat{\mathbf{y}}^{(1,p)}$
Σ
$\mathbf{v}_{cyc}^{(2)}$
$\mathbf{v}^{(3)} = \mathbf{v}_{cyc}^{(2)}$
$\mathbf{v}_{tre}^{(3)}$
$\mathbf{s}^{(1,3)} = \mathbf{v}_{tre}^{(3)}$
$\mathbf{W}_{in}^{(1,3)}$
$\mathbf{W}_{res}^{(1,3)}$
$\mathbf{W}_{pca}^{(1,3)}$
$\mathbf{W}_{out}^{(1,3)}$
$\mathbf{v}_{cyc}^{(N_D)}$
$\mathbf{s}^{(1,N_D+1)} = \mathbf{v}_{cyc}^{(N_D)}$
$\mathbf{W}_{in}^{(1,N_D+1)}$
$\mathbf{W}_{res}^{(1,N_D+1)}$
$\mathbf{W}_{pca}^{(1,N_D+1)}$
$\mathbf{W}_{out}^{(1,N_D+1)}$

Fig. 3. The original input time series data is recursively decomposed into N_D trend components and one cycle component. The (N_D+1) reservoir state matrices are obtained from the (N_D+1) reservoir encoders with $\left\{\mathbf{W}_{in}^{(1,1)}, \mathbf{W}_{res}^{(1,1)}\right\}$, $\left\{\mathbf{W}_{in}^{(1,2)}, \mathbf{W}_{res}^{(1,2)}\right\}$, ..., and $\left\{\mathbf{W}_{in}^{(1,N_D+1)}, \mathbf{W}_{res}^{(1,N_D+1)}\right\}$ directly (not shown) or through optional processing with the PCAs with mapping weights $\mathbf{W}_{pca}^{(1,1)}$, $\mathbf{W}_{pca}^{(1,2)}$, ..., and $\mathbf{W}_{pca}^{(1,N_D+1)}$. The model outputs are produced as the sum of the reservoir states weighted by $\mathbf{W}_{out}^{(1,1)}$, $\mathbf{W}_{out}^{(1,2)}$, ..., and $\mathbf{W}_{out}^{(1,N_D+1)}$.

Property (ESP) for each reservoir, we rescale $\mathbf{W}_{res}^{(1,p)}$ to satisfy the following condition [34]:

$$\rho\left((1-a)\,\mathbf{E} + a\mathbf{W}_{res}^{(1,p)}\right) < 1, \tag{2}$$

where $\rho(\cdot)$ represents a spectral radius and $\mathbf{E}$ is the identity matrix. After the above-mentioned initial settings, $\Theta_{enc}^{(1,p)} = \left\{\mathbf{W}_{in}^{(1,p)}, \mathbf{W}_{res}^{(1,p)}\right\}$ is kept fixed for $p = 1, 2, \ldots, N_P$ in the whole training and testing processes.

In the GroupedESN, the only one decoder, Θ_{dec}, is adjusted to map a collected state vector $\mathbf{c}(t)$ at time t to a desired output, where $\mathbf{c}(t) = \left[\mathbf{x}^{(1,1)}(t); \mathbf{x}^{(1,2)}(t); \ldots; \mathbf{x}^{(1,N_P)}(t)\right] \in \mathbb{R}^{N_P N_R}$. The output weight matrix of Θ_{dec}, $\mathbf{W}_{out}$, is optimized by minimizing a loss function based on the ridge regression [35] as follows:

$$\hat{\mathbf{W}}_{out} = \underset{\mathbf{W}_{out}}{\arg\min} \frac{1}{2N_T} \sum_{t=1}^{N_T} \|\mathbf{W}_{out}\mathbf{c}(t) - \mathbf{y}(t)\|_2^2 + \beta \|\mathbf{W}_{out}\|_F^2, \tag{3}$$

where parameter β is the Tikhonov regularization factor [36].

3. The proposed model: HP-MRESN

A schematic diagram of executing time-series prediction with our proposed HP-MRESN is shown in Fig. 3. The time series decomposer based on the HP filter with specific smoothing factors is used to recursively decompose a univariate input time series into multiple time series. Each decomposed time series is fed into the reservoir state extractor for generating high-dimensional temporal features. The Principal Component Analysis (PCA) [37] is optionally leveraged to reduce the redundant information in the generated reservoir states. The predicted outputs are obtained by the ensemble decoder. The following sections are devoted to the details of the three main parts of the HP-MRESN: the time series decomposer, the reservoir state extractor, and the ensemble decoder.

3.1. Time series decomposer

Typically, the HP filter is used to decompose one-dimensional time-series data into two components: a trend component (low-frequency signals) and a cycle component (high-frequency signals). In the proposed decomposer, we leverage HP filters to decompose the original input time series data into multiple time series components in a recursive manner. We assume that there are a total of N_D decompositions. We denote the one-dimensional time-series pending data at the d-th decomposition ($d = 1, 2, \ldots, N_D$) by

$$\mathbf{v}^{(d)} = [v^{(d)}(1), v^{(d)}(2), \ldots, v^{(d)}(N_T)]^\top. \tag{4}$$

This pending data is decomposed as follows:

$$\mathbf{v}^{(d)} = \mathbf{v}_{tre}^{(d)} + \mathbf{v}_{cyc}^{(d)}, \tag{5}$$

where the trend component $\mathbf{v}_{tre}^{(d)}$ and the cycle component $\mathbf{v}_{cyc}^{(d)}$ at the d-th decomposition are represented as follows:

$$\mathbf{v}_{tre}^{(d)} = \left[v_{tre}^{(d)}(1), v_{tre}^{(d)}(2), \ldots, v_{tre}^{(d)}(N_T)\right]^\top, \tag{6a}$$

$$\mathbf{v}_{cyc}^{(d)} = \left[v_{cyc}^{(d)}(1), v_{cyc}^{(d)}(2), \ldots, v_{cyc}^{(d)}(N_T)\right]^\top. \tag{6b}$$

Since the cycle component becomes the pending data at the subsequent decomposition, the pending data $\mathbf{v}^{(d)}$ is recursively given as follows:

$$\mathbf{v}^{(d)} = \begin{cases} \mathbf{u} & \text{if } d = 1 \\ \mathbf{v}_{cyc}^{(d-1)} & \text{if } 2 \le d \le N_D, \end{cases} \tag{7}$$

where $\mathbf{u} = [u(1), u(2), \ldots, u(N_T)]^\top$ represents the original univariate input time-series vector. The d-th decomposition by the HP filter is characterized by a minimization of the objective function given as follows [30]:

$$L^{(d)} = \phi^{(d)} \sum_{t=3}^{N_T} \left(v_{tre}^{(d)}(t) - 2v_{tre}^{(d)}(t-1) + v_{tre}^{(d)}(t-2)\right)^2$$

$$+\sum_{t=1}^{N_T}\left(v^{(d)}(t)-v_{tre}^{(d)}(t)\right)^2, \tag{8}$$

where $\phi^{(d)}$ is a positive smoothing parameter at the d-th decomposition. The unique optimal solution minimizing the above objective function is obtained as follows:

$$\mathbf{v}_{tre}^{(d)} = (\mathbf{E}+\phi^{(d)}\mathbf{B})^{-1}\mathbf{v}^{(d)}, \tag{9}$$

where $\mathbf{E} \in \mathbb{R}^{N_T\times N_T}$ is the identity matrix and $\mathbf{B} \in \mathbb{R}^{N_T\times N_T}$ is the band matrix given by

$$\mathbf{B} = \begin{pmatrix} 1 & -2 & 1 & \dots & 0 & 0 \\ -2 & 4+1 & -2-2 & \dots & 0 & 0 \\ 1 & -2-2 & 1+4+1 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1+4 & -2 \\ 0 & 0 & 0 & \dots & -2 & 1 \end{pmatrix}. \tag{10}$$

There is only one hyperparameter $\phi^{(d)}$ in each decomposition as seen in Eq. (9). For instance, the value of this smoothing parameter is set at 1600 for removing short-term fluctuations from quarterly business cycle data in economics [30]. To our best knowledge, there is no established method to appropriately choose the value of $\phi^{(d)}$ for time-series prediction with a given time-series dataset. We analyze the effects of the smoothing parameter value on the prediction performances of our proposed model in Section 5.4.2. Moreover, we provide an experimental analysis to show the validity of recursively decomposing the cycle component in Appendix C.

3.2. Reservoir state extractor

Inspired by the architecture of the GroupedESN introduced in Section 2.2, we employ multiple encoders placed in parallel to process the decomposed data mentioned in Section 3.1. Unlike the data feeding strategy of the GroupedESN, each encoder independently receives the corresponding decomposed one-dimensional time series as input data. The (N_D+1) time series given to the (N_D+1) encoders are represented as $\mathbf{s}^{(1,p)} = \left[s^{(1,p)}(1), s^{(1,p)}(2), \dots, s^{(1,p)}(N_T)\right]$ for $p = 1, 2, \dots, N_D+1$, where

$$s^{(1,p)}(t) = \begin{cases} v_{tre}^{(d)}(t), & \text{for } p = d = 1, 2, \dots, N_D \\ v_{cyc}^{(N_D)}(t), & \text{for } p = N_D+1. \end{cases} \tag{11}$$

For each parallel encoder, the corresponding reservoir states can be calculated with Eq. (1). We use the PCA as an optional operation to reduce redundant information in $\mathbf{x}^{(1,p)}(t)$. We define the reservoir state matrix including N_T reservoir state vectors of the encoder $\Theta_{enc}^{(1,p)}$ as $\mathbf{X}^{(1,p)} \in \mathbb{R}^{N_R\times N_T}$. We denote the mean of N_T reservoir state vectors corresponding to the p-th encoder by $\bar{\mathbf{x}}^{(1,p)}$ and obtain the mean-removed reservoir state matrix, $\tilde{\mathbf{X}}^{(1,p)} = [\tilde{\mathbf{x}}^{(1,p)}(1), \tilde{\mathbf{x}}^{(1,p)}(2), \dots, \tilde{\mathbf{x}}^{(1,p)}(N_T)] \in \mathbb{R}^{N_R\times N_T}$, where $\tilde{\mathbf{x}}^{(1,p)}(t) = \mathbf{x}^{(1,p)}(t) - \bar{\mathbf{x}}^{(1,p)}$. We assume that the dimension of the reservoir state is reduced to N_H by the PCA. We can obtain the mapping weights in the PCA for $\Theta_{enc}^{(1,p)}$, $\mathbf{W}_{pca}^{(1,p)} \in \mathbb{R}^{N_H\times N_R}$, by minimizing the reconstruction error as follows:

$$\hat{\mathbf{W}}_{pca}^{(1,p)} = \underset{\mathbf{W}_{pca}^{(1,p)}}{\arg\min} \left\| \tilde{\mathbf{X}}^{(1,p)} - \left(\mathbf{W}_{pca}^{(1,p)}\right)^\top \mathbf{W}_{pca}^{(1,p)} \tilde{\mathbf{X}}^{(1,p)} \right\|_F^2$$
$$\text{s.t. } \mathbf{W}_{pca}^{(1,p)}\left(\mathbf{W}_{pca}^{(1,p)}\right)^\top = \mathbf{E}, \tag{12}$$

where $\mathbf{E}$ is the $N_H \times N_H$ identity matrix. The solution is the principal eigenvectors of $\tilde{\mathbf{X}}^{(1,p)}(\tilde{\mathbf{X}}^{(1,p)})^\top$ corresponding to the first N_H principal eigenvalues in the descending order. The principal reservoir state component vector at time t is represented as follows:

$$\mathbf{x}_{pca}^{(1,p)}(t) = \hat{\mathbf{W}}_{pca}^{(1,p)}\mathbf{x}^{(1,p)}(t). \tag{13}$$

3.3. Ensemble decoder

We leverage an ensemble mode to obtain the final predicted outputs $\hat{\mathbf{y}}$ at time t by summing the partial predicted outputs as follows:

$$\hat{\mathbf{y}}(t) = \sum_{p=1}^{N_P}\hat{\mathbf{y}}^{(1,p)}(t). \tag{14}$$

The partial predicted outputs corresponding to $\Theta_{dec}^{(1,p)}$ at time t, $\hat{\mathbf{y}}^{(1,p)}(t)$, is given as the weighted sum of the collected state vector as follows:

$$\hat{\mathbf{y}}^{(1,p)}(t) = \mathbf{W}_{out}^{(1,p)}\mathbf{c}^{(1,p)}(t), \tag{15}$$

where $\mathbf{c}^{(1,p)}(t) = \mathbf{x}_{pca}^{(1,p)}(t)$ if we apply the PCA as shown in Eq. (13), otherwise $\mathbf{c}^{(1,p)}(t) = \mathbf{x}^{(1,p)}(t)$. We can obtain an optimal $\mathbf{W}_{out}^{(1,p)}$ by minimizing the loss function as follows:

$$\hat{\mathbf{W}}_{out}^{(1,p)} = \underset{\mathbf{W}_{out}^{(1,p)}}{\arg\min}\, \frac{1}{2N_T}\sum_{t=1}^{N_T}\left\|\mathbf{W}_{out}^{(1,p)}\mathbf{c}^{(1,p)}(t)-\mathbf{y}(t)\right\|_2^2 + \beta\left\|\mathbf{W}_{out}^{(1,p)}\right\|_F^2. \tag{16}$$

We can obtain the closed-form solution as follows:

$$\hat{\mathbf{W}}_{out}^{(1,p)} = \mathbf{Y}^{(1,p)}\mathbf{X}^{\mathrm{T}}\left(\mathbf{C}^{(1,p)}(\mathbf{C}^{(1,p)})^{\mathrm{T}}+\beta\mathbf{E}\right)^{-1}, \tag{17}$$

where $\mathbf{Y}^{(1,p)} = [\mathbf{y}^{(1,p)}(1), \mathbf{y}^{(1,p)}(2), \dots, \mathbf{y}^{(1,p)}(N_T)] \in \mathbb{R}^{N_Y\times N_T}$ is the target collection matrix and $\mathbf{C}^{(1,p)} = \left[\mathbf{c}^{(1,p)}(1), \mathbf{c}^{(1,p)}(2), \dots, \mathbf{c}^{(1,p)}(N_T)\right] \in \mathbb{R}^{N_C\times N_T}$ is the corresponding state collection matrix.

4. Hyperparameter search and computational complexity

From the descriptions in Section 3, it is obvious that the number of decompositions N_D directly determines the architecture of our proposed model. We show main steps of a greedy method specialized for building a well-trained HP-MRESN in Section 4.1, which can automatically find the most suitable value of N_D for maximizing the predictive performance, and we give an analysis of the corresponding computational complexity in Section 4.2.

4.1. Pseudocode

In prediction tasks considered later, a time series data is separated into initial transient, training, validation, and testing sets in this order. We denote the maximum time index for the transient, training, and validation sets by t_{ini}, t_{tr}, and t_{val}, respectively. An example of separating a time series into the above-mentioned four sets is shown in Fig. 4. For simplicity, we consider a situation that we have an original univariate input series $\mathbf{u}$, a training target set $\mathbf{Y}_{tr} = [\mathbf{y}(t_{ini}+1), \mathbf{y}(t_{ini}+2), \dots, \mathbf{y}(t_{tr})] \in \mathbb{R}^{N_Y\times(t_{tr}-t_{ini})}$, and a validation target set $\mathbf{Y}_{val} = [\mathbf{y}(t_{tr}+1), \mathbf{y}(t_{tr}+2), \dots, \mathbf{y}(t_{val})] \in \mathbb{R}^{N_Y\times(t_{val}-t_{tr})}$. We define N_D^{max} as the maximal number of decompositions. The pseudocode of the greedy algorithm for architecture optimization in the HP-MRESN is described in Algorithm 1.

In Algorithm 1, the reservoir state matrix corresponding to the training set is represented as $\mathbf{X}_{tr}^{(1,p)} = [\mathbf{x}^{(1,p)}(t_{ini}+1), \mathbf{x}^{(1,p)}(t_{ini}+2), \dots, \mathbf{x}^{(1,p)}(t_{tr})] \in \mathbb{R}^{N_R\times(t_{tr}-t_{ini})}$ and the reservoir state matrix corresponding to the validation set $\mathbf{X}_{tr}^{(1,p)} = [\mathbf{x}^{(1,p)}(t_{tr}+1), \mathbf{x}^{(1,p)}(t_{tr}+2), \dots, \mathbf{x}^{(1,p)}(t_{val})] \in \mathbb{R}^{N_R\times(t_{val}-t_{tr})}$. The upper limit of decompositions in the search process is given by N_D^{max}. In procedures 5–10, we recursively decompose the pending data into a trend part $\mathbf{v}_{tre}^{(d)}$ and a cycle part $\mathbf{v}_{cyc}^{(d)}$ (corresponding to Section 3.1). In procedure 11, we generate two reservoir state matrices, $\mathbf{X}^{(1,d)}$ and $\mathbf{X}^{(1,d+1)}$, by encoding $\mathbf{s}^{(d)}$ and $\mathbf{s}^{(d+1)}$ (corresponding to Section 3.2). In procedures 11–14, we train the d-th and $(d+1)$-th

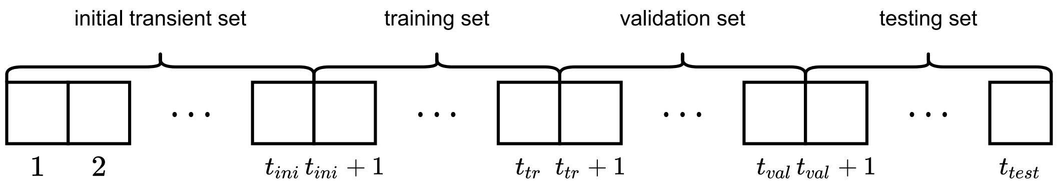

Fig. 4. An example of separating a time series into initial transient, training, validation, and testing sets.

Algorithm 1: The greedy algorithm for optimizing the architecture of the HP-MRESN

```
Data: original input series $\mathbf{u}$, targets for training set $\mathbf{Y}_{tr}$, targets for validation set $\mathbf{Y}_{val}$, spectral radius $\rho$, input scaling $\delta$, reservoir size $N_R$, leaking rate $\alpha$, sparsity $\eta$, regularization factor $\beta$, smoothing factors $\phi^{(d)}$ for $d = 1, 2, \dots, N_D^{max}$
Result: optimal number of decomposition $N_D$, parallel encoders $\Theta_{enc}^{(1,1)}, \Theta_{enc}^{(1,2)}, \dots, \Theta_{enc}^{(1,N_D+1)}$, decoders $\Theta_{dec}^{(1,1)}, \Theta_{dec}^{(1,2)}, \dots, \Theta_{dec}^{(1,N_D+1)}$
1  init
2      $\hat{\mathbf{Y}}_{temp}, \mathbf{Y}_{pre} \leftarrow \mathbf{0}$;
3      $\Theta_{enc}^{(1,1)}, \Theta_{enc}^{(1,2)}, \dots, \Theta_{enc}^{(1,N_D^{max}+1)} \leftarrow$ PrepareEncoders($N_D^{max}, \rho, \alpha, \delta, \eta$);
4  for $d \leftarrow 1$ to $N_D^{max}$ do
5      if $d == 1$ then
6          $\mathbf{v}^{(d)} \leftarrow \mathbf{u}$;
7      else
8          $\mathbf{v}^{(d)} \leftarrow \mathbf{v}_{cyc}^{(d-1)}$;
9      end
10     $\mathbf{v}_{tre}^{(d)}, \mathbf{v}_{cyc}^{(d)} \leftarrow$ Decompose($\phi^{(d)}, \mathbf{v}^{(d)}$);
11     $\mathbf{s}^{(1,d)} \leftarrow \mathbf{v}_{tre}^{(d)}$ and $\mathbf{s}^{(1,d+1)} \leftarrow \mathbf{v}_{cyc}^{(d)}$;
12     $\mathbf{X}^{(1,d)} \leftarrow$ Encode($\Theta_{enc}^{(1,d)}, \mathbf{s}^{(1,d)}$) and $\mathbf{X}^{(1,d+1)} \leftarrow$ Encode($\Theta_{enc}^{(1,d+1)}, \mathbf{s}^{(1,d+1)}$);
13     $\Theta_{dec}^{(1,d)} \leftarrow$ TrainDecoder($\mathbf{X}_{tr}^{(1,d)}, \mathbf{Y}_{tr}^{(1,d)}, \beta$);
14     $\Theta_{dec}^{(1,d+1)} \leftarrow$ TrainDecoder($\mathbf{X}_{tr}^{(1,d+1)}, \mathbf{Y}_{tr}^{(1,d+1)}, \beta$);
15     $\hat{\mathbf{Y}}_{val}^{(1,d)} \leftarrow$ Decode($\Theta_{dec}^{(1,d)}, \mathbf{X}_{val}^{(1,d)}$) and $\hat{\mathbf{Y}}_{val}^{(1,d+1)} \leftarrow$ Decode($\Theta_{dec}^{(1,d+1)}, \mathbf{X}_{val}^{(1,d+1)}$);
16     $\hat{\mathbf{Y}}_{temp} \leftarrow \hat{\mathbf{Y}}_{temp} + \hat{\mathbf{Y}}_{val}^{(1,d)}$ and $\hat{\mathbf{Y}}_{val} \leftarrow \hat{\mathbf{Y}}_{temp} + \hat{\mathbf{Y}}_{val}^{(1,d+1)}$;
17     if NRMSE($\mathbf{Y}_{pre}, \mathbf{Y}_{val}$) $\geq$ NRMSE($\hat{\mathbf{Y}}_{val}, \mathbf{Y}_{val}$) or $d == 1$ then
18         $\mathbf{Y}_{pre} \leftarrow \hat{\mathbf{Y}}_{val}$;
19     else
20         $N_D \leftarrow d - 1$;
21         break;
22     end
23 end
24 return $\Theta_{enc}^{(1,1)}, \Theta_{enc}^{(1,2)}, \dots, \Theta_{enc}^{(1,N_D+1)}, \Theta_{dec}^{(1,1)}, \Theta_{dec}^{(1,2)}, \dots, \Theta_{dec}^{(1,N_D+1)}$, and $N_D$.
```

decoders (corresponding to Eq. (14) in Section 3.3). The predicted values corresponding to the validation set, $\hat{\mathbf{y}}$, are calculated in procedures 15–16 (corresponding to Eq. (15) in Section 3.3). By checking the condition described in procedure 17, we decide whether a better value of N_D yielding a smaller prediction error and the corresponding architecture of HP-MRESN are found or not. For simplicity, we omitted the training procedures of the PCA [37].

4.2. Analyses of computational complexity

Based on the algorithm shown in Section 4.1, we analyze the corresponding computational complexity. The computational complexity of each decomposition, $\mathcal{C}_{HP}$, can be calculated as follows:

$$\mathcal{C}_{HP} = \mathcal{O}\left(N_T^2\right). \tag{18}$$

The computational complexity of the process at each encoder, $\mathcal{C}_{enc}$, can be expressed as follows:

$$\mathcal{C}_{enc} = \mathcal{O}\left(N_T N_R + N_T N_R^2\right). \tag{19}$$

The computational complexity of PCA, $\mathcal{C}_{PCA}$, can be represented as follows:

$$\mathcal{C}_{PCA} = \mathcal{O}\left(\min\left(N_T^3, N_R^3\right)\right). \tag{20}$$

The computational complexity for solving the closed-form equation in each decoder, $\mathcal{C}_{dec}$, can be described as follows:

$$\mathcal{C}_{dec} = \mathcal{O}\left(N_T N_H^2 + N_H^3 + N_Y N_H^2\right). \tag{21}$$

Note that $N_H \leq N_R \ll N_T$, $N_U \ll N_H$, and $N_Y \ll N_H$ in many NTP tasks. Under this condition, the total computational complexity of the proposed HP-MRESN with N_D decompositions can be summarized as follows:

$$\begin{aligned}\mathcal{C}_{total} &= (N_D)\,\mathcal{C}_{HP} + (N_D+1)\,(\mathcal{C}_{enc} + \mathcal{C}_{PCA} + \mathcal{C}_{dec}) \\ &= \mathcal{O}\left(\left(N_D N_T^2\right) + (N_D+1)\left(N_T N_R^2 + N_R^3 + N_T\,(N_H)^2\right)\right) \\ &\approx \mathcal{O}\left(N_D N_T^2 + (N_D+1)\,N_T N_R^2\right) \\ &\approx \mathcal{O}\left(N_D N_T^2 + N_D N_T N_R^2\right).\end{aligned} \tag{22}$$

If $N_T < N_R^2$ as shown in Tables 1 and 2, we can further simplify the computational complexity to $\mathcal{C}_{total} \approx \mathcal{O}\left(N_D N_T N_R^2\right)$, which indicates that the computational complexity of the HP filters, PCAs, and decoders can be ignored. For a comparison of the computational complexity between the proposed model and the three representative MRESN models shown in Fig. 2, we assume that the number of reservoirs in the DeepESN and the MSLESN is N_L and the number of them in the GroupedESN is N_P. Based on these assumptions, Ref. [14] estimates the computational complexity of the DeepESN at $\mathcal{O}\left(N_L^2 N_T N_R^2\right)$, and Ref. [15] estimates that of the MSLESN at $\mathcal{O}\left(N_L N_T N_R^2\right)$. Based on the architecture of GroupedESN, we can derive the computational complexity of the GroupedESN as $\mathcal{O}\left(N_P^2 N_T N_R^2\right)$. In most cases, N_L, N_P, and N_D are much smaller than N_T and N_R. If the number of reservoirs is the same (i.e. $N_D = N_P = N_L$), then we can conclude that the computational complexity of the proposed model is the same as that of the MSLESN and is smaller than those of the DeepESN and the GroupedESN.

The essential reason for the lower computational complexity of the proposed HP-MRESN is that it does not use a large reservoir state collection matrix by concatenating the states from multiple reservoirs unlike DeepESN and GroupedESN. It should be noted that the time complexity for the computation in Eq. (17) is proportional to the order of the square of the number of columns of

Table 1
Data partition for six time-series datasets.

	Initial transient	Training	Validation	Testing
Cardio	131	500	200	200
Sunspot	250	2000	500	500
DMTMA	112	3000	1000	1000
Electricity	182	2200	1000	1000
Bike	100	2900	1000	1000
Traffic speed	783	4000	2000	2000

Table 2
Parameter settings common to all the tested models.

Parameters	Symbol	Value
Reservoir size	N_R	[100, 200, ..., 500]
Leaking rate	α	[0.1, 0.2, ..., 1]
Input scaling	δ	[0.001, 0.01, 0.1, 1]
Maximal number of decompositions	N_D^{max}	10
Sparsity of reservoir connection	η	90%
Spectral radius	ρ	0.95
Regularization factor	β	1E−06

the reservoir state collection matrix. An additional reason is that the computational complexity for the HP filter and PCA parts in the proposed model can be negligible.

Practical analyses of computation time for model training will be shown in Section 5.4.4, which demonstrate the computational efficiency of our proposed model in practice.

5. Numerical simulations

In this section, we report the details and results of our experiments for evaluating the prediction performance of the HP-MRESN in comparison with other relevant models. Specifically, six real-world time-series datasets and their partitions are introduced in Section 5.1, the evaluation metric for computational ability is described in Section 5.2, and the baseline models and experimental settings are given in Section 5.3. The simulation results for all the comparison models are presented in Section 5.4.

5.1. Dataset description and task settings

In this study, we executed performance comparison in various prediction tasks with six real-world time-series datasets. The brief information of these datasets are given as follows:

- Cardio: a dataset of daily number of cardiovascular inpatients in Hong Kong hospitals [38].
- Sunspot: jerky monthly sunspot number dataset from January, 1749 to November, 2019 [39].
- DMTMA: daily maximum temperature in Melbourne airport from January 1st, 2006 to December 31st, 2019 [40].
- Electricity: daily electricity consumption of open power system data in Germany, from January 1st, 2006 to December 31st, 2017 [41].
- Bike: total number of rental bikes per hour from 21:00, May 9th, 2011 to 0:00, December 5th, 2011 [42].
- Traffic speed: urban traffic speed per 10 min dataset at the No. 61 road segment in Guangzhou from August 1st, 2016 to September 30th, 2016 [43].

We rescaled the values of the six datasets into the range of [0,1], examples of which are shown in Fig. 5. The lengths of the initial transient set, the training set, the validation set, and the testing set for the six datasets are listed in Table 1.

5.2. Evaluation metrics

The Normalized Root Mean Square Error (NRMSE) is used for evaluating the prediction performance, which is defined as follows:

$$\mathrm{NRMSE} = \frac{\sqrt{\frac{1}{N_T}\sum_{t=1}^{N_T}\left(\hat{\mathbf{y}}(t)-\mathbf{y}(t)\right)^2}}{\sqrt{\frac{1}{N_T}\sum_{t=1}^{N_T}\left(\mathbf{y}(t)-\bar{\mathbf{y}}\right)^2}}, \tag{23}$$

where $\mathbf{y}(t)$ is the t-th target output, $\hat{\mathbf{y}}(t)$ is the t-th predicted output, and $\bar{\mathbf{y}}$ denotes the mean of N_T data values of $\mathbf{y}(t)$ from $t = 1$ to N_T.

5.3. Experimental settings and models for comparison

Since the PCA is an optional process in the HP-MRESN, we separately denote the models with PCA and without PCA by HP-MRESN-PCA and HP-MRESN, respectively, in the following experiments. For the HP-MRESN-PCA, we set the retained dimension after the PCA process, N_H, at the minimal value satisfying the following condition:

$$\frac{\sum_{i=1}^{N_H}\lambda_i}{\sum_{i=1}^{N_R}\lambda_i} \geq \mu, \tag{24}$$

where N_R eigenvalues of the variance–covariance matrix (see Eq. (12)) are sorted in a descending order, i.e., $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_{N_R}$. The range of the ratio μ is $0 < \mu \leq 1$. In the experiments, we searched the suitable value of μ in $[0.1, 0.2, \ldots, 1]$ for each specific task.

We designed two schemes called the "equal scheme" and the "descending scheme" in the assignment of $\phi^{(d)}$ for $d = 1, 2, \ldots, N_D^{max}$. In the equal scheme, we set $\phi^{(1)} = \phi^{(2)} = \cdots = \phi^{(N_D^{max})}$ and searched the values of $\phi^{(d)}$ for $d = 1, 2, \ldots, N_D^{max}$ in $[1, 10, 20, 50, 100]$. In the descending scheme, we assigned the values of smoothing parameters $\phi^{(1)}, \phi^{(2)}, \ldots, \phi^{(N_D^{max})}$ as $\phi^{(1)} = N_D^{max}, \phi^{(2)} = N_D^{max}-1, \ldots,$ and $\phi^{(N_D^{max})} = 1$. The fixed and searched values of the other model parameters are listed in Table 2. We employed a grid search to determine the best parameter settings and computed the average performance over 20 independent trials for each parameter setting.

To investigate the effectiveness of the proposed model, we compared its prediction performance with that of the other MRESN models, including the DeepESN [14], the MSLESN [15], the GroupedESN [17], the Broad-ESN [23], the WESN [27], and the EEMD-ESN [25]. We searched an optimal number of layers, N_L, for the DeepESN and MSLESN in the range of [1, 2, ...,11]. The Broad-ESN, WESN and EEMD-ESN share the same scheme of "decomposition-(parallel)encoding–decoding" with the proposed HP-MRESN. For the settings of the Restricted Boltzmann Machine (RBM) [44] embedded into the Broad-ESN, we searched an optimal dimension of the hidden layer in the range of [1, 2, ...,11] and used contrastive divergence with five iterations (CD-5) reported in [24] to tune the hidden weight matrix. For the prepossessing based on the Discrete Wavelet Transform (DWT) [26] in the WESN, we used the orthonormal wavelet basis function reported in Ref. [27]. For the EEMD-ESN, we strictly followed the settings reported in Ref. [25] where the number of trials for the Empirical Mode Decomposition (EMD) [45] with added noise was kept at 100, the standard deviation of Gaussian noise was fixed at 0.5, and the sliding window and the polynomial order of Savitzky–Golay (SG) filtering [29] were set at nine and four, respectively.

Since previous studies show that prediction performances of single-reservoir ESNs are not competitive with those of MRESNs in various time-series prediction tasks [14,15,40], we exclude the single-reservoir ESNs from the performance comparison in Section 5.4. Nevertheless, we provide the prediction performance of the standard ESN as a reference in Appendix B.

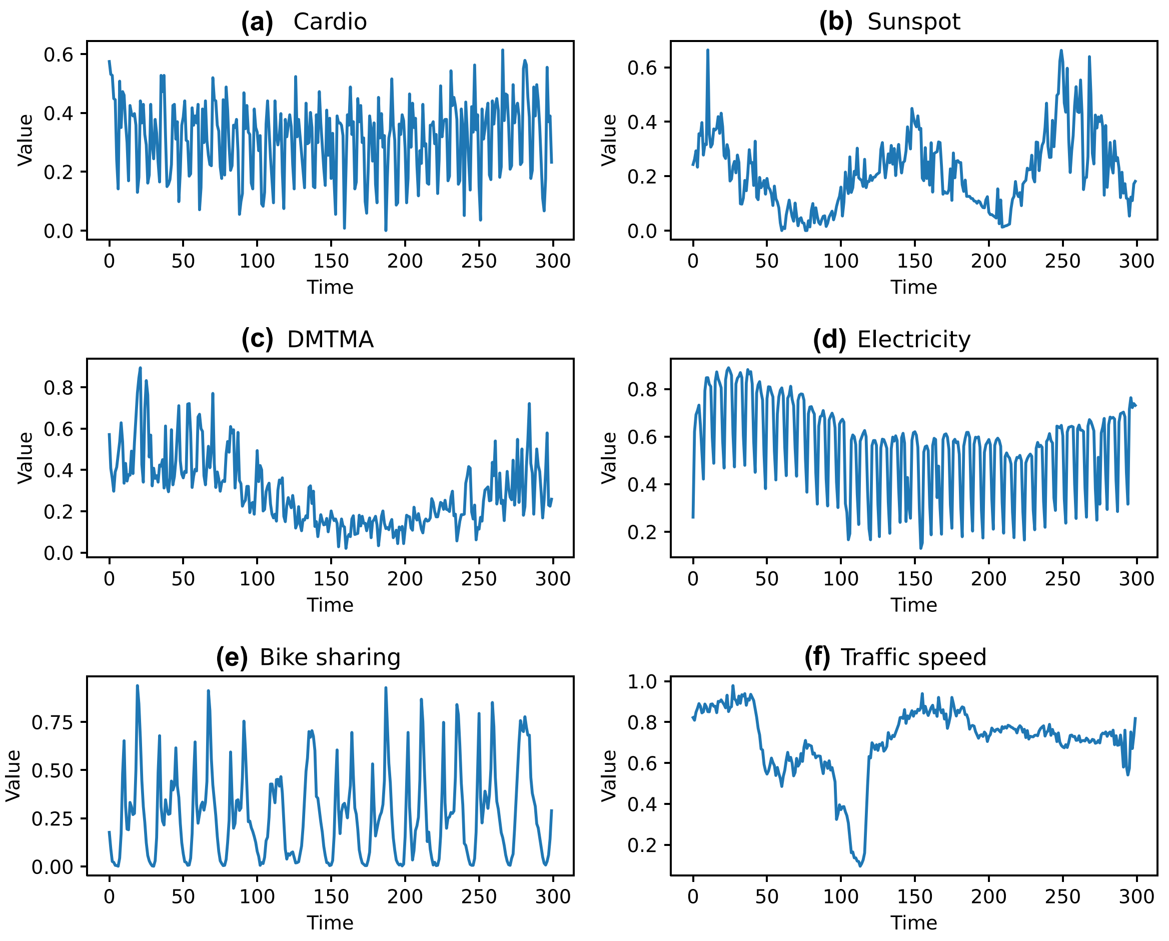

Fig. 5. Example display of six datasets. (a) Cardio, (b) Sunspot, (c) DMTMA, (d) Electricity, (e) Bike, and (f) Traffic speed.

Table 3
Average NRMSEs ± (std) for all the tested models on the Cardio dataset.

	$K=1$	$K=3$	$K=5$	Multi-output
DeepESN	6.47E−01 ± (5.04E−03)	7.28E−01 ± (7.80E−03)	7.33E−01 ± (1.16E−02)	7.30E−01 ± (2.62E−03)
MSLESN	6.41E−01 ± (3.63E−03)	7.20E−01 ± (1.51E−02)	7.27E−01 ± (8.98E−03)	7.18E−01 ± (3.32E−03)
GroupedESN	6.40E−01 ± (4.33E−03)	7.27E−01 ± (5.04E−03)	7.36E−01 ± (1.23E−02)	7.27E−01 ± (3.99E−03)
Broad-ESN	6.46E−01 ± (8.25E−03)	7.33E−01 ± (4.27E−03)	7.48E−01 ± (3.56E−03)	7.36E−01 ± (8.24E−03)
WESN	4.51E−01 ± (4.47E−03)	5.51E−01 ± (1.07E−02)	6.04E−01 ± (1.13E−02)	5.64E−01 ± (7.02E−02)
EEMD-ESN	3.46E−01 ± (2.50E−03)	4.35E−01 ± (4.51E−03)	**5.68E−01** ± (**1.89E−03**)	6.40E−01 ± (1.14E−03)
HP-MRESN	8.97E−02 ± (1.01E−03)	4.19E−01 ± (6.22E−03)	5.86E−01 ± (4.25E−03)	6.03E−01 ± (5.24E−03)
HP-MRESN-PCA	**8.88E−02** ± **(1.08E−03)**	**4.17E−01** ± **(6.25E−03)**	5.82E−01 ± (8.78E−03)	**5.11E−01** ± **(6.17E−03)**

5.4. Simulation results

5.4.1. Prediction performance of the HP-MRESN and other MRESN models

We numerically evaluated computational performance of the proposed HP-MRESN in four settings of time series prediction tasks with the above-mentioned six real-world time-series datasets. Three of them are K-step-ahead prediction tasks for $K = 1, 3$, and 5, where the target output is given as $\mathbf{y}(t) = \mathbf{u}(t+K)$. The other one is a multi-output prediction task, where the target data is formulated as $\mathbf{y}(t) = [\mathbf{u}(t+1), \mathbf{u}(t+3), \mathbf{u}(t+5)]^\top$. Note that the last task was considered for testing the ability of producing accurate multiple outputs based on the reservoir states under the best parameter settings in a one-shot manner, which is a more challenging task than the other three tasks. In summary, totally 24 tasks were considered for each of the proposed and the other tested models.

The prediction results are listed in Tables 3–8 corresponding to the six time series datasets. In the one-step-ahead and the three-step-ahead prediction tasks with all the datasets, we observe that the HP-MRESN and HP-MRESN-PCA outperform most of the other models. In the five-step-ahead prediction, the HP-MRESN has a slight advantage only for the Electricity dataset. In the multi-output prediction tasks with all the datasets, the HP-MRESN-PCA has the smallest NRMSE. Moreover, the MRESN models following the scheme of "decomposition-(parallel)encoding–decoding" (i.e. the WESN, EEMD-ESN, HP-MRESN and HP-MRESN-PCA) have smaller prediction errors than the other MRESN models in all the tasks.

To justify the significant differences between the proposed model and the other tested models, we performed statistical analyses on the prediction performances. Since all the reported results are normal and homoscedastic, we used the one-way analysis of variance (ANOVA) [46] to determine whether there are any significant differences between the mean values of NRMSEs of the tested methods for the 24 prediction tasks. Following the

Table 4
Average NRMSEs ± (std) for all the tested models on the Sunspot dataset.

	$K = 1$	$K = 3$	$K = 5$	Multi-output
DeepESN	3.19E−01 ± (6.56E−04)	3.64E−01 ± (1.32E−03)	3.82E−01 ± (1.96E−03)	3.55E−01 ± (1.79E−03)
MSLESN	3.16E−01 ± (8.65E−04)	3.63E−01 ± (1.56E−03)	3.83E−01 ± (3.29E−03)	3.57E−01 ± (1.93E−03)
GroupedESN	3.20E−01 ± (3.82E−04)	3.63E−01 ± (1.08E−03)	3.82E−01 ± (2.44E−03)	3.52E−01 ± (6.12E−04)
BroadESN	3.15E−01 ± (2.37E−04)	3.61E−01 ± (8.48E−04)	3.82E−01 ± (1.64E−03)	3.53E−01 ± (1.33E−03)
WESN	1.95E−01 ± (1.46E−03)	2.89E−01 ± (1.63E−03)	3.18E−01 ± (5.19E−03)	2.74E−01 ± (2.52E−03)
EEMD-ESN	1.78E−01 ± (1.94E−04)	**1.90E−01 ± (2.95E−04)**	**2.20E−01 ± (2.83E−04)**	1.97E−01 ± (1.71E−04)
HP-MRESN	4.11E−02 ± (3.25E−04)	1.95E−01 ± (9.17E−04)	2.71E−01 ± (2.01E−03)	1.71E−01 ± (7.32E−04)
HP-MRESN-PCA	**4.09E−02 ± (3.19E−04)**	1.95E−01 ± (5.89E−04)	2.70E−01 ± (8.83E−04)	**1.69E−01 ± (1.32E−03)**

Table 5
Average NRMSEs ± (std) for all the tested models on the DMTMA dataset.

	$K = 1$	$K = 3$	$K = 5$	Multi-output
DeepESN	5.92E−01 ± (7.53E−04)	6.68E−01 ± (1.80E−03)	6.72E−01 ± (4.87E−04)	6.51E−01 ± (6.44E−04)
MSLESN	5.94E−01 ± (2.28E−03)	6.69E−01 ± (8.24E−04)	6.67E−01 ± (2.13E−03)	6.54E−01 ± (9.23E−04)
GroupedESN	5.94E−01 ± (2.28E−03)	6.68E−01 ± (1.80E−03)	6.67E−01 ± (2.13E−03)	6.52E−01 ± (2.58E−04)
BroadESN	5.86E−01 ± (7.85E−04)	6.69E−01 ± (9.48E−04)	6.78E−01 ± (1.32E−03)	6.52E−01 ± (8.97E−04)
WESN	3.49E−01 ± (1.56E−03)	5.32E−01 ± (1.58E−03)	5.96E−01 ± (1.28E−03)	5.06E−01 ± (1.91E−03)
EEMD-ESN	3.39E−01 ± (2.83E−04)	3.83E−01 ± (1.04E−03)	**4.40E−01 ± (7.90E−04)**	4.04E−01 ± (5.69E−04)
HP-MRESN	7.94E−02 ± (6.76E−04)	**3.45E−01 ± (1.04E−03)**	4.94E−01 ± (1.90E−03)	3.66E−01 ± (2.06E−03)
HP-MRESN-PCA	**7.90E−02 ± (9.24E−04)**	3.46E−01 ± (7.53E−04)	4.92E−01 ± (1.37E−03)	**3.65E−01 ± (1.77E−03)**

Table 6
Average NRMSEs ± (std) for all the tested models on the Electricity dataset.

	$K = 1$	$K = 3$	$K = 5$	Multi-output
DeepESN	2.79E−01 ± (3.80E−03)	3.70E−01 ± (3.30E−03)	4.11E−01 ± (3.48E−03)	3.65E−01 ± (2.91E−03)
MSLESN	2.68E−01 ± (3.00E−03)	3.60E−01 ± (6.11E−03)	4.11E−01 ± (4.61E−03)	3.64E−01 ± (3.65E−03)
GroupedESN	2.90E−01 ± (2.50E−02)	3.67E−01 ± (5.01E−03)	4.14E−01 ± (4.40E−03)	3.58E−01 ± (2.62E−03)
BroadESN	2.96E−01 ± (4.71E−03)	3.80E−01 ± (3.95E−03)	4.21E−01 ± (4.15E−03)	3.76E−01 ± (6.12E−03)
WESN	2.06E−01 ± (4.96E−03)	3.28E−01 ± (4.77E−03)	3.57E−01 ± (2.12E−03)	3.11E−01 ± (1.37E−03)
EEMD-ESN	4.24E−01 ± (1.67E−03)	4.41E−01 ± (7.29E−04)	4.53E−01 ± (7.41E−04)	4.47E−01 ± (7.45E−04)
HP-MRESN	4.84E−02 ± (4.94E−04)	**1.72E−01 ± (1.34E−03)**	**2.67E−01 ± (1.25E−03)**	1.79E−01 ± (9.30E−04)
HP-MRESN-PCA	**4.83E−02 ± (6.69E−04)**	1.72E−01 ± (1.80E−03)	2.67E−01 ± (1.87E−03)	**1.78E−01 ± (1.11E−03)**

Table 7
Average NRMSEs ± (std) for all the tested models on the Bike dataset.

	$K = 1$	$K = 3$	$K = 5$	Multi-output
DeepESN	3.02E−01 ± (9.10E−03)	5.43E−01 ± (2.11E−02)	6.30E−01 ± (3.15E−02)	5.19E−01 ± (1.61E−02)
MSLESN	2.68E−01 ± (4.91E−03)	5.29E−01 ± (1.95E−02)	7.06E−01 ± (1.04E−02)	4.97E−01 ± (1.76E−02)
GroupedESN	3.03E−01 ± (6.42E−03)	5.29E−01 ± (1.95E−02)	6.34E−01 ± (3.95E−02)	5.23E−01 ± (1.63E−02)
BroadESN	3.01E−01 ± (5.18E−03)	5.52E−01 ± (9.30E−03)	6.02E−01 ± (9.35E−03)	4.85E−01 ± (1.12E−02)
WESN	2.78E−01 ± (8.18E−03)	6.95E−01 ± (4.53E−02)	7.89E−01 ± (5.83E−02)	6.20E−01 ± (4.37E−02)
EEMD-ESN	2.08E−01 ± (8.83E−04)	2.44E−01 ± (7.73E−04)	**3.03E−01 ± (1.09E−03)**	2.53E−01 ± (6.50E−04)
HP-MRESN	3.94E−02 ± (5.09E−04)	1.70E−01 ± (2.26E−03)	3.29E−01 ± (1.51E−02)	1.80E−01 ± (5.63E−03)
HP-MRESN-PCA	**3.89E−02 ± (7.42E−04)**	**1.65E−01 ± (2.63E−03)**	3.25E−01 ± (1.03E−02)	**1.68E−01 ± (3.30E−03)**

Table 8
Average NRMSEs ± (std) for all the tested models on the Traffic speed dataset.

	$K = 1$	$K = 3$	$K = 5$	Multi-output
DeepESN	2.31E−01 ± (2.77E−04)	3.43E−01 ± (2.16E−03)	4.02E−01 ± (5.06E−03)	3.57E−01 ± (3.02E−03)
MSLESN	2.32E−01 ± (5.58E−04)	3.51E−01 ± (2.39E−04)	4.38E−01 ± (7.10E−03)	3.53E−01 ± (1.50E−03)
GroupedESN	2.31E−01 ± (6.92E−04)	3.55E−01 ± (4.42E−03)	4.26E−01 ± (7.67E−03)	3.56E−01 ± (1.20E−03)
BroadESN	2.28E−01 ± (1.14E−03)	3.43E−01 ± (3.43E−03)	4.11E−01 ± (6.07E−03)	3.47E−01 ± (2.40E−03)
WESN	1.34E−01 ± (1.63E−03)	2.54E−01 ± (2.40E−03)	3.39E−01 ± (6.48E−03)	2.53E−01 ± (2.13E−03)
EEMD-ESN	1.09E−01 ± (1.04E−04)	**1.18E−01 ± (1.02E−04)**	**1.49E−01 ± (1.19E−04)**	1.71E−01 ± (1.49E−04)
HP-MRESN	2.92E−02 ± (2.20E−04)	1.31E−01 ± (5.56E−04)	2.30E−01 ± (4.64E−03)	1.68E−01 ± (6.56E−04)
HP-MRESN-PCA	**2.90E−02 ± (2.78E−04)**	1.31E−01 ± (6.69E−04)	2.25E−01 ± (2.29E−03)	**1.67E−01 ± (6.44E−04)**

definitions of Omnibus test [46], we treated the eight models as eight groups and regarded the corresponding 24 prediction results as the observed variables in each group. If the calculated p-value corresponding to the Omnibus test is larger than a designated significance level, it indicates that the differences between the mean values of NRMSEs of the tested models for the 24 prediction tasks are significant. Then, we used the post-hoc Tukey's Honestly Significant Difference (Tukey's HSD) test [47] to infer which differences are significant. We set the corresponding significance level at 0.05.

The means (M) and the standard deviations (SD) of NRMSEs of the tested models in the 24 prediction tasks are reported in Table 9. We also reported that the p-value of the one-way ANOVA test obtained with Autorank [48] is 3.03E−40 ≈ 0. Based on these results, we rejected the null hypothesis ($p = 0.000$) of the one-way ANOVA test assuming that there is no difference between

Table 9
The means (M) and the corresponding standard deviations (SD) of NRMSEs of the tested models for the 24 prediction tasks.

	DeepESN	MSLESN	GroupedESN	Broad-ESN	WESN	EEMD-ESN	HP-MRESN	HP-MRESN-PCA
M	0.483	0.483	0.484	0.483	0.408	0.319	0.233	0.228
SD	0.165	0.167	0.163	0.165	0.176	0.145	0.166	0.158

Table 10
Best parameter settings for HP-MRESN and HP-MRESN-PCA on six real-world datasets.

	$K = 1$		$K = 3$	
	HP-MRESN	HP-MRESN-PCA	HP-MRESN	HP-MRESN-PCA
Cardio	$N_R = 500$, $\alpha = 0.8$, $\delta = 0.1$, $N_D = 10$	$N_R = 500$, $\alpha = 0.8$, $\delta = 0.1$, $N_D = 10$, $\mu = 0.7$	$N_R = 500$, $\alpha = 0.8$, $\delta = 0.1$, $N_D = 10$	$N_R = 500$, $\alpha = 0.8$, $\delta = 0.1$, $N_D = 10$, $\mu = 0.8$
Sunspot	$N_R = 500$, $\alpha = 0.3$, $\delta = 1$, $N_D = 10$	$N_R = 500$, $\alpha = 0.3$, $\delta = 1$, $N_D = 10$, $\mu = 0.2$	$N_R = 400$, $\alpha = 0.7$, $\delta = 0.1$, $N_D = 10$	$N_R = 400$, $\alpha = 0.7$, $\delta = 0.1$, $N_D = 10$, $\mu = 0.1$
DMTMA	$N_R = 500$, $\alpha = 0.3$, $\delta = 1$, $N_D = 10$	$N_R = 500$, $\alpha = 0.3$, $\delta = 1$, $N_D = 10$, $\mu = 0.3$	$N_R = 500$, $\alpha = 0.3$, $\delta = 1$, $N_D = 10$	$N_R = 500$, $\alpha = 0.3$, $\delta = 1$, $N_D = 10$, $\mu = 0.2$
Electricity	$N_R = 500$, $\alpha = 1$, $\delta = 0.1$, $N_D = 10$	$N_R = 500$, $\alpha = 1$, $\delta = 0.1$, $N_D = 10$, $\mu = 1$	$N_R = 500$, $\alpha = 0.8$, $\delta = 1$, $N_D = 10$	$N_R = 500$, $\alpha = 0.8$, $\delta = 1$, $N_D = 10$, $\mu = 1$
Bike	$N_R = 500$, $\alpha = 0.4$, $\delta = 1$, $N_D = 10$	$N_R = 500$, $\alpha = 0.5$, $\delta = 1$, $N_D = 10$, $\mu = 0.5$	$N_R = 500$, $\alpha = 0.7$, $\delta = 1$, $N_D = 10$	$N_R = 500$, $\alpha = 0.8$, $\delta = 1$, $N_D = 10$, $\mu = 1$
Traffic speed	$N_R = 500$, $\alpha = 0.4$, $\delta = 1$, $N_D = 10$	$N_R = 500$, $\alpha = 0.4$, $\delta = 1$, $N_D = 10$, $\mu = 0.3$	$N_R = 500$, $\alpha = 0.7$, $\delta = 0.1$, $N_D = 10$	$N_R = 500$, $\alpha = 0.6$, $\delta = 0.1$, $N_D = 10$, $\mu = 1$
	$K = 5$		Multi-output	
	HP-MRESN	HP-MRESN-PCA	HP-MRESN	HP-MRESN-PCA
Cardio	$N_R = 500$, $\alpha = 0.7$, $\delta = 0.1$, $N_D = 7$	$N_R = 500$, $\alpha = 0.9$, $\delta = 0.1$, $N_D = 7$, $\mu = 0.3$	$N_R = 500$, $\alpha = 0.8$, $\delta = 0.1$, $N_D = 10$	$N_R = 500$, $\alpha = 0.8$, $\delta = 0.1$, $N_D = 10$, $\mu = 0.6$
Sunspot	$N_R = 500$, $\alpha = 0.3$, $\delta = 0.1$, $N_D = 7$	$N_R = 500$, $\alpha = 0.3$, $\delta = 0.1$, $N_D = 7$, $\mu = 0.8$	$N_R = 500$, $\alpha = 0.4$, $\delta = 0.1$, $N_D = 10$	$N_R = 500$, $\alpha = 0.4$, $\delta = 1$, $N_D = 10$, $\mu = 0.1$
DMTMA	$N_R = 500$, $\alpha = 0.5$, $\delta = 0.1$, $N_D = 7$	$N_R = 500$, $\alpha = 0.4$, $\delta = 0.1$, $N_D = 7$, $\mu = 0.2$	$N_R = 500$, $\alpha = 0.3$, $\delta = 1$, $N_D = 10$	$N_R = 500$, $\alpha = 0.3$, $\delta = 1$, $N_D = 10$, $\mu = 0.2$
Electricity	$N_R = 500$, $\alpha = 0.7$, $\delta = 1$, $N_D = 7$	$N_R = 500$, $\alpha = 0.8$, $\delta = 1$, $N_D = 7$, $\mu = 0.8$	$N_R = 500$, $\alpha = 0.9$, $\delta = 1$, $N_D = 10$	$N_R = 500$, $\alpha = 0.9$, $\delta = 1$, $N_D = 10$, $\mu = 0.7$
Bike	$N_R = 500$, $\alpha = 0.7$, $\delta = 1$, $N_D = 8$	$N_R = 500$, $\alpha = 0.7$, $\delta = 1$, $N_D = 7$, $\mu = 0.5$	$N_R = 500$, $\alpha = 0.7$, $\delta = 1$, $N_D = 8$	$N_R = 500$, $\alpha = 1$, $\delta = 1$, $N_D = 8$, $\mu = 0.8$
Traffic speed	$N_R = 500$, $\alpha = 0.5$, $\delta = 0.1$, $N_D = 7$	$N_R = 500$, $\alpha = 0.4$, $\delta = 0.1$, $N_D = 7$, $\mu = 0.1$	$N_R = 500$, $\alpha = 0.3$, $\delta = 1$, $N_D = 10$	$N_R = 500$, $\alpha = 0.3$, $\delta = 1$, $N_D = 10$, $\mu = 0.6$

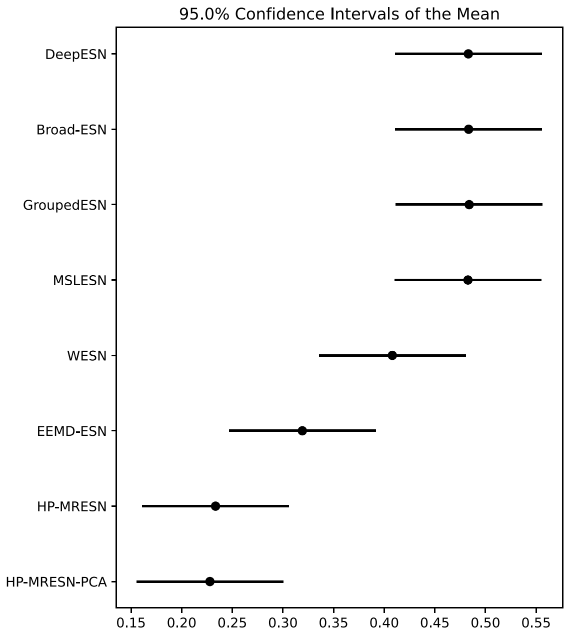

Fig. 6. Confidence intervals of all the tested models in the Tukey's HSD test.

the mean values of NRMSEs of the tested methods over the 24 prediction tasks. The confidence intervals based on the Tukey's HSD test for all the tested methods are shown in Fig. 6, where we can clearly find that the performances of the HP-MRESN and HP-MRESN-PCA are significantly better than those of the WESN, MSLESN, Broad-ESN, DeepESN and GroupedESN.

The best parameter settings of the HP-MRESN and HP-MRESN-PCA on the 24 tasks with six datasets are listed in Table 10. These parameter settings are found on the basis of the performance for the validation set in each task. We can see that the best value of N_R reaches 500 in most cases, which points out that an increase in the reservoir size is beneficial to enhance the representation ability of the encoder. The ratio μ is smaller than one in most cases and its average value over the total 24 tasks is 0.534, indicating that the PCA effectively reduces the redundancy of the collected state matrix in each decoder. We can also notice that the best number of decompositions is ten in most cases, which means that more decompositions can help the model generate better prediction results. We list the best hyperparameter settings of the Broad-ESN, WESN, and EEMD-ESN for the 24 tasks with the six datasets in Appendix A.

We report the average training times of all the tested models in the 24 tasks with six datasets in Fig. 7. For a fair comparison, we set $N_R = 500$ for all the models. Every model was set to have 11 reservoir state encoders. The value of N_D for the HP-MRESN, HP-MRESN-PCA, WESN, EEMD-ESN was kept at 10. The size of hidden layer in the RBM of the Broad-ESN was set at 11. We executed experiments with a computer having Intel (R) Core i9-7900X CPU and 96 GB RAM. All the models were implemented with Pytorch v1.10.2 [49]. The HP filter in our proposed models was implemented with Statsmodels v0.12.2 [50]. The Ensemble Empirical Mode Decomposition (EEMD) and the Savitzky–Golay filter (SG filter) used in the EEMD-ESN were implemented with PyEMD [51] and SciPy v1.7.1 [52], respectively.

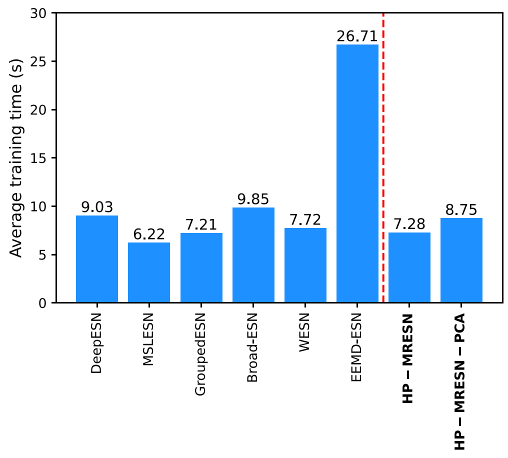

Fig. 7. Average training time over 20 trials for all the tested models in the 24 tasks with six datasets. For a fair comparison of average training times, N_R was kept at 500 for all the tested models.

The Discrete Wavelet Transform (DWT) was implemented with PyWavelets [53] in the WESN. The average training time of the HP-MRESN was 7.28 s, which is only slightly longer than that of the MSLESN. In addition, it is noticeable that the training time of the HP-MRESN is almost the same as that of the GroupedESN, which indicates that the computational cost of the HP filters is small enough to be ignored for the data length treated in the experiments. Due to the optimization of the PCA projection weights, the training time of the HP-MRESN-PCA is a little longer than that of the HP-MRESN but still shorter than those of the DeepESN, Broad-ESN, and EEMD-ESN. We notice that the EEMD-ESN takes the longest time among all the tested models, which is caused by heavy computations for the EEMD and SG filter. The above analyses experimentally demonstrate that our choice of the HP filter for time series decomposition is reasonable from the viewpoint of computational efficiency.

5.4.2. *Effect of smoothing schemes in the HP-MRESN*

Fig. 8 shows prediction performances of the HP-MRESN with the parameter settings listed in Table 10 under different schemes about the smoothing factors $\phi^{(d)}$ in the multi-output prediction task with the six datasets. We observe that the descending scheme enables the HP-MRESN to reach the lowest NRMSE for each dataset. Although smoothing factor values in the equal scheme, including $\phi^{(d)} = 1$, 10 and 20, can also lead to relatively low prediction errors comparable to those in the descending scheme for some datasets when the number of decompositions is

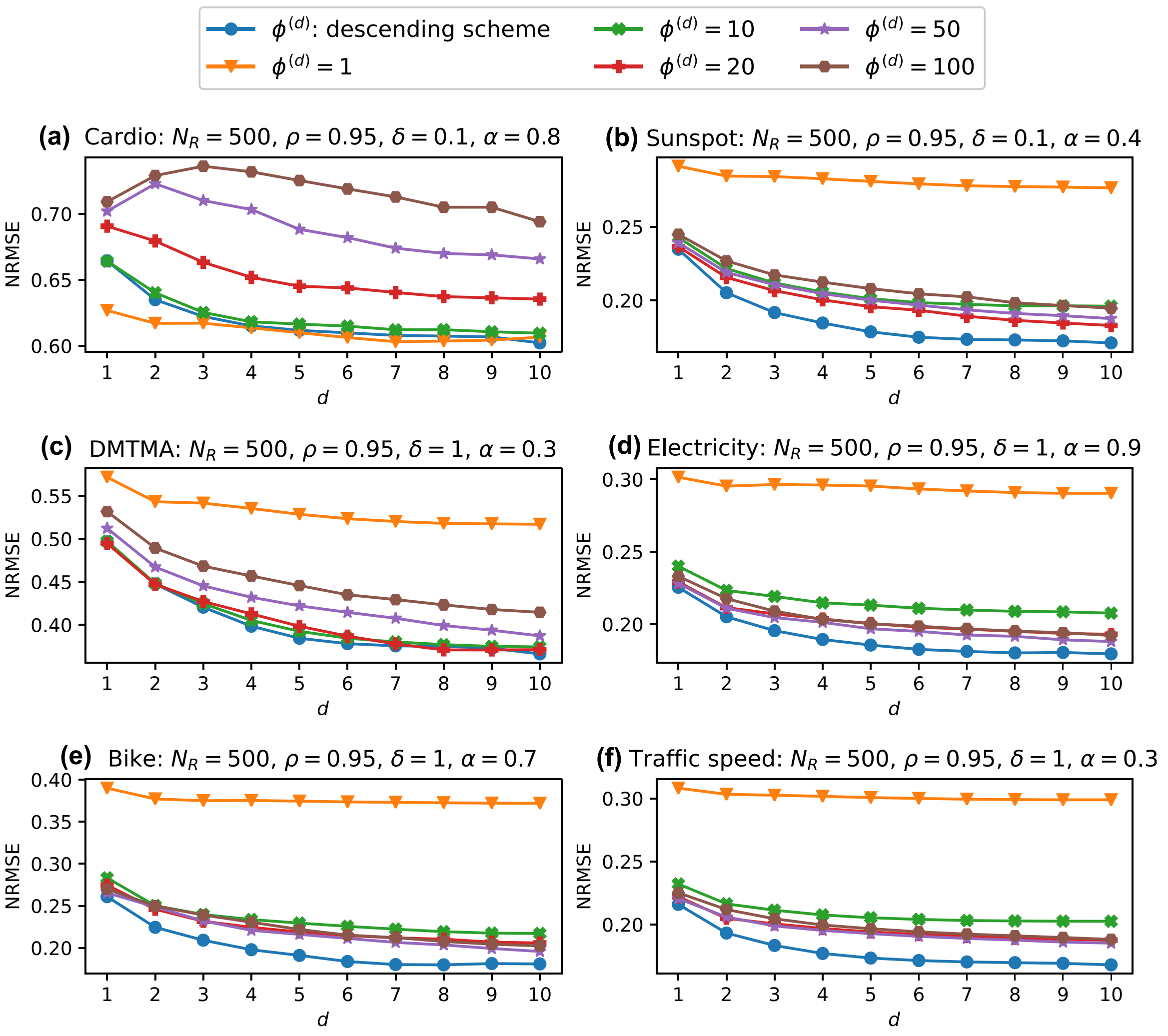

Fig. 8. Prediction performances of the HP-MRESN with a variation of the index of decomposition from one to ten under various smoothing factor values in the multi-output prediction task with six datasets. (a) Cardio, (b) Sunspot, (c) DMTMA, (d) Electricity, (e) Bike, and (f) Traffic speed.

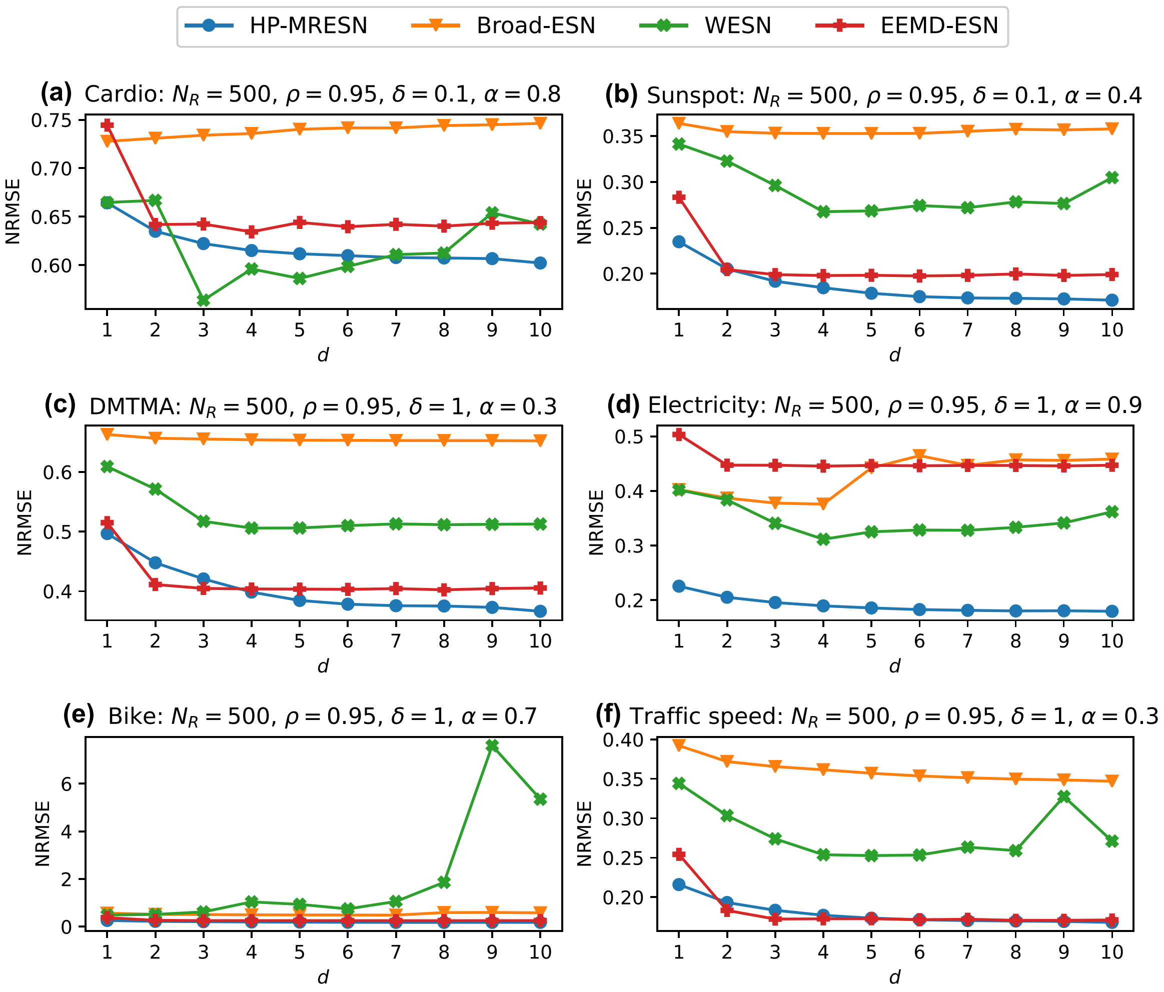

Fig. 9. Prediction performances of the HP-MRESN, Broad-ESN, WESN, and EEMD-ESN with a variation of d from one to ten in the multi-output prediction task with six real-world datasets. (a) Cardio, (b) Sunspot, (c) DMTMA, (d) Electricity, (e) Bike, and (f) Traffic speed.

sufficiently large, these three smoothing factor values cannot ensure relatively stable prediction performances for all the datasets. Therefore, we showed the best prediction performances of our proposed model with the descending scheme.

5.4.3. Comparison with Broad-ESN, WESN, and EEMD-ESN

The HP-MRESN shares the model architecture based on "decomposition-(parallel)encoding–decoding" with the Broad-ESN, WESN, and EEMD-ESN. We examined the influences of the different time-series decomposers used for these models. In this comparison, we show prediction performances of the models under the best parameter settings described in Section 5.4.1 and Appendix A in the multi-output prediction task with six time-series datasets.

The prediction performances of the HP-MRESN, Broad-ESN, WESN, and EEMD-ESN when varying index d from one to ten are shown in Fig. 9. On the whole, we discover obvious decreases in the NRMSEs for the HP-MRESN with the descending scheme when increasing d from one to six, and slight decreases when increasing d from seven to ten on the five datasets (except Bike dataset). For the EEMD-ESN, we find that the obvious decreasing trends of NRMSEs appear only when varying d from one to two. For the WESN, it is clear that the decreases in NRMSEs are relatively distinct when varying d from one to three but the subsequent prediction performances become gradually unstable and even worse for larger d.

5.4.4. Comparison between HP-MRESN and fully-trained RNN models

We compared the best performances of the HP-MRESN with those of the Elman network (ElmanNet) [4], the Long Short Term Memory (LSTM) [6], and the Gated Recurrent Unit (GRU) [7] in the multi-output prediction tasks with six datasets. These three fully-trained RNN models were implemented with the same computer as that described in Section 5.4.1. Moreover, to test whether fully-trained RNN models work well in the architecture of "decomposition-(parallel)encoding–decoding", we combined the HP filter with these models, which we call HP-ElmanNet, HP-LSTM, and HP-GRU. For fair comparisons, we configured 500 hidden units for every fully-trained RNN model. The hyperbolic tangent function was adopted as the nonlinear activation function. We used the Adam [54] as the optimization algorithm. We set the maximal training epochs at 1000 and fixed the learning rate at 0.001. For the HP-MRESN, we adopted prediction performance of a well-trained model with Algorithm 1. Each of the HP-ElmanNet, HP-LSTM, and HP-GRU has the same number of decompositions and encoders as the HP-MRESN.

The average performances and training times are shown in Fig. 10. We can clearly observe that the HP-MRESN has lower prediction errors and training time than the fully-trained RNN-based models. The ElmanNet takes relatively low training time but the corresponding prediction performances are the worst among all

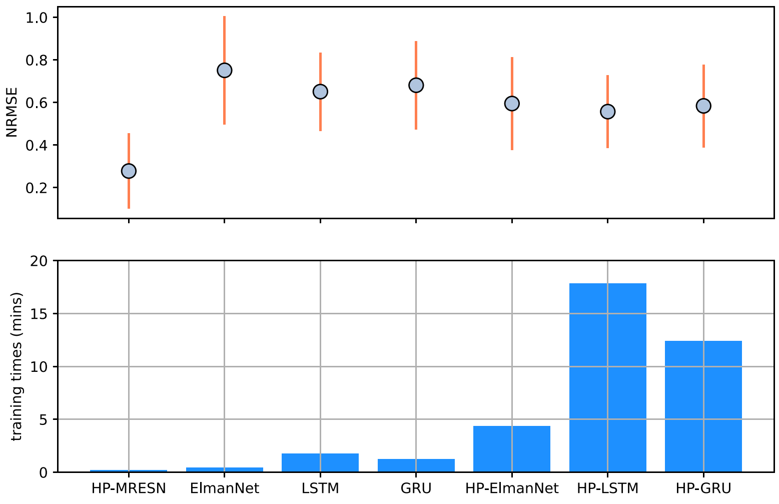

Fig. 10. Average prediction performances (upper) and training times (lower) of the HP-MRESN, the ElmanNet, the LSTM, the GRU, the HP-ElmanNet, the HP-LSTM, and the HP-GRU.

the models. Even though the LSTM can obtain the lowest error among the original fully-trained RNN models, the corresponding training time is around 14.3 times as long as that of the HP-MRESN. On the other hand, we can notice that the NRMSEs of the HP-ElmanNet, HP-LSTM, and HP-GRU are on average 20.07% lower than those of the ElmanNet, LSTM, and GRU. However, the corresponding training times are around ten times higher than those of the original fully-trained RNN models, because the optimal N_D found through the architecture optimization algorithm for each dataset is around ten.

6. Discussion

In this study, we proposed the novel MRESN model called HP-MRESN composed of an HP-based time-series decomposer, a reservoir state extractor, and an ensemble decoder. To show the computational ability of our proposed model for real-world NTP tasks, we evaluated prediction performance of the HP-MRESN and compared it with state-of-the-art MRESN-based models and fully-trained RNN models for the total of 24 tasks with six real-world datasets. Comparative results and corresponding statistical analyses shown in Section 5.4.1 indicate that the proposed HP-MRESN can outperform other state-of-the-art MRESN-based models in many NTP tasks. Subsequently, the comparison of training times among MRESN-based models revealed that the practical training time of the HP-MRESN is almost the same as the most efficient existing MRESN-based models. Moreover, the comparative results shown in Section 5.4.4 highlight the superiority of the HP-MRESN over the fully-trained RNN models in the aspects of prediction performance and training cost. In addition, we provided prediction performance of the HP-MRESN for four classical chaotic time series datasets in Appendix D. These results demonstrate high computational effectiveness and efficiency of our proposed model in NTP tasks with chaotic time series data.

In terms of the scheme for assigning smoothing factors in the HP-filter-based decomposer, the experimental results in Section 5.4.2 indicate that the descending scheme is a better choice than the equal scheme for tackling with NTP tasks. The proposed descending scheme was empirically derived from our previous study suggesting that the smoothing factor of each decomposition should be less than or equal to ten [55]. On the other hand, in comparison with the other models combining other filtering methods with multiple ESNs [22,23,25], prediction results in Section 5.4.3 suggest that the HP filter is a better choice as a prepossessing method for complex time-series input data in real-world NTP tasks.

A significance of this paper based on the above conclusions is that MRESN-based models following the architecture of "decomposition-(parallel)encoding–decoding" as in our proposed HP-MRESN obviously outperform other types of MRESN-based models on the NTP tasks. This finding not only points out a direction for designing more effective MRESN-based models in prediction tasks, but also indicates a potential of them for handling other time-series processing tasks such as time-series anomaly detection.

Some related studies [14,17,23–25] only compare the prediction performances of their proposed MRESN-based models with those of ESN models with a single reservoir, and experimentally demonstrate their superiority in different NTP tasks. To our best knowledge, few studies have systematically evaluated the performance of various MRESN-based models in NTP tasks. Therefore, another significance of this paper is the comprehensive comparison of prediction performances among many MRESN-based models. Specifically, this study focuses on evaluating MRESN-based models having multiple parallel reservoir encoders, which complements the one-sided knowledge drawn from a recent study mainly focusing on an evaluation of multiple layered reservoir encoders [56].

Although the proposed HP-MRESN with the descending scheme for setting the smoothing factors works well in this study, a further investigation is needed to clarify a relationship between the number of decompositions and the assignment scheme for the smoothing factor in each decomposition. Moreover, since the HP filter is a powerful and efficient tool mainly used for removing trend movements in the business cycle, there is no detailed analysis about why the HP filter works well in dealing with NTP tasks. Further theoretical analysis is another future work. In this study, for simplicity, we fixed some hyparameters (e.g., the spectral radius and the maximal number of decompositions) of the HP-MRESN. By using some advanced heuristic approaches, such as Particle Swarm Optimization [57] and Bayesian Optimization [58], for flexibly searching hyperparameters in the model depending on the task, the prediction performance could be improved in exchange for increased computational efforts. Although we concentrated on comparing prediction performances of RNN-based models and MRESN models in this work, further performance

Table A.1
The best parameter settings for Broad-ESN, WESN, and EEMD-ESN in the 24 tasks with six real-world datasets.

	$K = 1$			$K = 3$		
	Broad-ESN	WESN	EEMD-ESN	Broad-ESN	WESN	EEMD-ESN
Cardio	$N_R = 300$, $\alpha = 1$, $\delta = 0.01$, $N_D = 7$	$N_R = 400$, $\alpha = 1$, $\delta = 0.01$, $N_D = 2$	$N_R = 300$, $\alpha = 0.8$, $\delta = 0.1$, $N_D = 4$	$N_R = 200$, $\alpha = 1$, $\delta = 0.01$, $N_D = 6$	$N_R = 300$, $\alpha = 1$, $\delta = 0.001$, $N_D = 3$	$N_R = 500$, $\alpha = 0.9$, $\delta = 0.01$, $N_D = 6$
Sunspot	$N_R = 400$, $\alpha = 0.5$, $\delta = 0.01$, $N_D = 8$	$N_R = 400$, $\alpha = 0.9$, $\delta = 0.01$, $N_D = 3$	$N_R = 500$, $\alpha = 0.2$, $\delta = 0.1$, $N_D = 1$	$N_R = 500$, $\alpha = 0.7$, $\delta = 0.01$, $N_D = 9$	$N_R = 500$, $\alpha = 0.4$, $\delta = 0.01$, $N_D = 4$	$N_R = 500$, $\alpha = 1$, $\delta = 0.1$, $N_D = 8$
DMTMA	$N_R = 400$, $\alpha = 0.1$, $\delta = 1$, $N_D = 7$	$N_R = 500$, $\alpha = 1$, $\delta = 0.01$, $N_D = 4$	$N_R = 200$, $\alpha = 0.3$, $\delta = 0.1$, $N_D = 1$	$N_R = 100$, $\alpha = 0.1$, $\delta = 0.1$, $N_D = 10$	$N_R = 500$, $\alpha = 0.7$, $\delta = 0.01$, $N_D = 4$	$N_R = 200$, $\alpha = 0.8$, $\delta = 1$, $N_D = 3$
Electricity	$N_R = 200$, $\alpha = 1$, $\delta = 1$, $N_D = 10$	$N_R = 200$, $\alpha = 1$, $\delta = 0.1$, $N_D = 4$	$N_R = 500$, $\alpha = 1$, $\delta = 0.01$, $N_D = 1$	$N_R = 200$, $\alpha = 0.9$, $\delta = 1$, $N_D = 10$	$N_R = 200$, $\alpha = 1$, $\delta = 0.1$, $N_D = 4$	$N_R = 500$, $\alpha = 1$, $\delta = 0.1$, $N_D = 6$
Bike	$N_R = 300$, $\alpha = 1$, $\delta = 1$, $N_D = 9$	$N_R = 300$, $\alpha = 1$, $\delta = 0.1$, $N_D = 3$	$N_R = 200$, $\alpha = 0.4$, $\delta = 0.01$, $N_D = 1$	$N_R = 500$, $\alpha = 0.5$, $\delta = 1$, $N_D = 5$	$N_R = 300$, $\alpha = 0.7$, $\delta = 1$, $N_D = 3$	$N_R = 500$, $\alpha = 1$, $\delta = 1$, $N_D = 4$
Traffic speed	$N_R = 500$, $\alpha = 0.2$, $\delta = 0.1$, $N_D = 10$	$N_R = 300$, $\alpha = 1$, $\delta = 0.01$, $N_D = 3$	$N_R = 300$, $\alpha = 0.3$, $\delta = 0.1$, $N_D = 1$	$N_R = 400$, $\alpha = 0.3$, $\delta = 0.1$, $N_D = 10$	$N_R = 500$, $\alpha = 1$, $\delta = 0.01$, $N_D = 4$	$N_R = 500$, $\alpha = 1$, $\delta = 1$, $N_D = 4$
	$K = 5$			Multi-output		
	Broad-ESN	WESN	EEMD-ESN	Broad-ESN	WESN	EEMD-ESN
Cardio	$N_R = 300$, $\alpha = 1$, $\delta = 0.1$, $N_D = 2$	$N_R = 400$, $\alpha = 0.8$, $\delta = 0.001$, $N_D = 6$	$N_R = 500$, $\alpha = 1$, $\delta = 0.1$, $N_D = 4$	$N_R = 300$, $\alpha = 1$, $\delta = 0.01$, $N_D = 4$	$N_R = 500$, $\alpha = 0.9$, $\delta = 0.001$, $N_D = 3$	$N_R = 500$, $\alpha = 1$, $\delta = 0.1$, $N_D = 8$
Sunspot	$N_R = 500$, $\alpha = 0.4$, $\delta = 0.1$, $N_D = 3$	$N_R = 500$, $\alpha = 0.3$, $\delta = 0.01$, $N_D = 7$	$N_R = 500$, $\alpha = 1$, $\delta = 0.1$, $N_D = 5$	$N_R = 300$, $\alpha = 0.4$, $\delta = 0.1$, $N_D = 3$	$N_R = 500$, $\alpha = 0.4$, $\delta = 0.01$, $N_D = 6$	$N_R = 500$, $\alpha = 1$, $\delta = 0.1$, $N_D = 6$
DMTMA	$N_R = 100$, $\alpha = 0.1$, $\delta = 0.1$, $N_D = 10$	$N_R = 400$, $\alpha = 0.4$, $\delta = 0.001$, $N_D = 5$	$N_R = 400$, $\alpha = 1$, $\delta = 0.1$, $N_D = 5$	$N_R = 300$, $\alpha = 0.1$, $\delta = 0.1$, $N_D = 10$	$N_R = 400$, $\alpha = 0.7$, $\delta = 0.01$, $N_D = 5$	$N_R = 400$, $\alpha = 1$, $\delta = 0.1$, $N_D = 4$
Electricity	$N_R = 500$, $\alpha = 0.7$, $\delta = 1$, $N_D = 4$	$N_R = 500$, $\alpha = 0.9$, $\delta = 0.01$, $N_D = 4$	$N_R = 500$, $\alpha = 0.9$, $\delta = 1$, $N_D = 8$	$N_R = 500$, $\alpha = 1$, $\delta = 1$, $N_D = 4$	$N_R = 500$, $\alpha = 1$, $\delta = 0.01$, $N_D = 4$	$N_R = 500$, $\alpha = 1$, $\delta = 0.1$, $N_D = 3$
Bike	$N_R = 500$, $\alpha = 0.5$, $\delta = 1$, $N_D = 5$	$N_R = 200$, $\alpha = 0.7$, $\delta = 1$, $N_D = 3$	$N_R = 500$, $\alpha = 0.6$, $\delta = 1$, $N_D = 6$	$N_R = 400$, $\alpha = 0.7$, $\delta = 1$, $N_D = 7$	$N_R = 300$, $\alpha = 0.7$, $\delta = 1$, $N_D = 3$	$N_R = 500$, $\alpha = 1$, $\delta = 1$, $N_D = 6$
Traffic speed	$N_R = 500$, $\alpha = 0.3$, $\delta = 0.1$, $N_D = 10$	$N_R = 500$, $\alpha = 0.7$, $\delta = 0.01$, $N_D = 5$	$N_R = 500$, $\alpha = 1$, $\delta = 1$, $N_D = 5$	$N_R = 400$, $\alpha = 0.4$, $\delta = 0.1$, $N_D = 10$	$N_R = 500$, $\alpha = 1$, $\delta = 0.01$, $N_D = 5$	$N_R = 500$, $\alpha = 1$, $\delta = 0.1$, $N_D = 8$

comparisons with other types of neural networks such as fuzzy neural networks [59,60] remain to be addressed.

CRediT authorship contribution statement

Ziqiang Li: Conceptualization, Data curation, Methodology, Software, Validation, Writing – original draft. **Yun Liu:** Software, Validation, Writing – review & editing. **Gouhei Tanaka:** Methodology, Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

We have shared the link of the codes and datasets in the abstract part of this paper.

Acknowledgments

This work was partially supported by JST CREST, Japan Grant Number JPMJCR19K2 (ZL and GT), JSPS, Japan KAKENHI Grant Number 20K11882 (GT), Project of Intelligent Mobility Society Design, Japan, Social Cooperation Program, Japan, UTokyo (GT), Japan, AI Center, UTokyo (GT), Japan, and partially based on results obtained from Project No. JPNP16007, commissioned by the New Energy and Industrial Technology Development Organization (NEDO) (GT).

Appendix A. Hyperparameter settings

We provide the best hyperparameter settings for the Broad-ESN, WESN, and EEMD-ESN in Table A.1, which correspond to the results in Tables 3–8 for the 24 prediction tasks.

Appendix B. Prediction performances of the standard ESN

Table B.1 shows the average prediction performances of the standard ESN [34] for the 24 tasks with the six real-world time-series datasets described in Section 5.1. We evaluated the performances after optimizing the hyperparameter settings as described in Table 2. Comparing with the prediction errors of the MRESN-based models shown in Tables 3–8, we can obviously find that the prediction performance of the standard ESN is much worse than that of the MRESNs.

Appendix C. Validity of recursive cycle component decompositions in HP-MRESN

In HP-MRESN, we decompose cycle components recursively instead of trend components. We consider a variant model named HP-MRESN-V where the trend components are recursively decomposed. Reservoir states are generated from N_D cycle components and the last trend component in the HP-MRESN-V. The prediction performances of the HP-MRESN and the HP-MRESN-V for $d = 1, 2, \ldots, 10$ in the multi-output prediction tasks with six datasets are shown in Fig. C.1. There are growing gaps in the NRMSEs between the HP-MRESN and the HP-MRESN-V when $d > 2$. These experimental evidences indicate that recursively decomposing cycle components yields better prediction performances than recursively decomposing trend components. Specifically, we show a glimpse of the normalized Bike dataset in Fig. C.2(a) and the corresponding three trend and cycle components generated by the time series decomposer of the HP-MRESN in Fig. C.2(b). We can observe that the three trend components have less irregular fluctuations than the three cycle components. Fig. C.2(c) presents three violin plots of prediction errors in 20 random trials for these six components. We can find that both NRMSEs on trend and cycle components are decreasing for increasing d. After each decomposition, the errors on the trend components are much smaller than those on the cycle components, which means that

Table B.1
Average NRMSEs ± (std) for the standard ESN in the 24 tasks with six real-world time-series datasets.

	$K = 1$	$K = 3$	$K = 5$	Multi-output
Cardio	6.39E−01 ± (4.82E−03)	7.22E−01 ± (1.17E−02)	7.36E−01 ± (1.23E−02)	7.32E−01 ± (4.62E−03)
Sunspot	3.16E−01 ± (7.97E−04)	3.63E−01 ± (1.48E−03)	3.84E−01 ± (2.38E−03)	3.54E−01 ± (1.68E−03)
DMTMA	5.89E−01 ± (1.65E−03)	6.70E−01 ± (4.02E−03)	6.73E−01 ± (1.91E−03)	6.55E−01 ± (8.17E−04)
Electricity	2.80E−01 ± (3.19E−03)	3.64E−01 ± (4.45E−03)	4.12E−01 ± (5.07E−03)	3.67E−01 ± (3.36E−03)
Bike	2.86E−01 ± (4.68E−03)	5.20E−01 ± (1.16E−02)	6.33E−01 ± (3.17E−02)	5.13E−01 ± (1.21E−02)
Traffic	2.33E−01 ± (1.56E−03)	3.57E−01 ± (5.87E−03)	4.29E−01 ± (1.26E−02)	3.72E−01 ± (6.91E−03)

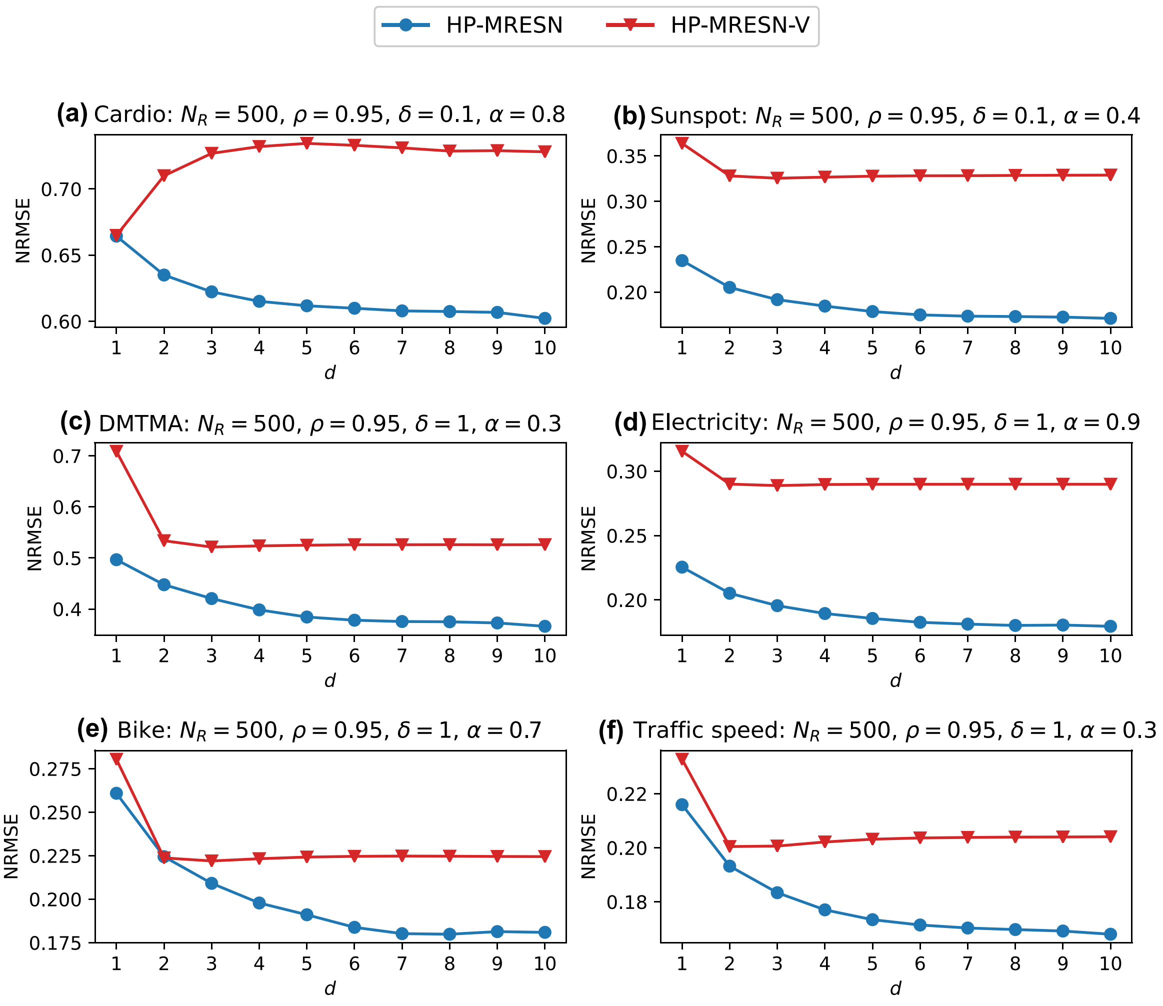

Fig. C.1. Prediction performances of HP-MRESN and HP-MRESN-V with a variation of d from one to ten in the multi-output prediction tasks with six datasets. (a) Cardio, (b) Sunspot, (c) DMTMA, (d) Electricity, (e) Bike, and (f) Traffic speed.

encoding temporal features of the trend components recursively separated from pending data in the HP-MRESN is a reasonable choice.

Appendix D. Supplementary experiments with benchmark chaotic time-series datasets

The Mackey–Glass System (MGS) [61] and the Sante Fe Laser (Laser) dataset [62] were used for evaluating the prediction performances of the proposed HP-MRESN. The MGS is formulated as follows:

$$y(t+1) = y(t) + \delta \left(a \frac{y(t-\varphi/\delta)}{1 + y(t-\varphi/\delta)^n} - by(t) \right), \qquad \text{(D.1)}$$

where parameters a, b, n, and δ are set at 0.2, −0.1, 10, and 0.1, respectively. The MGS produces chaotic time series data when φ is larger than 16.8. We set $\varphi = 17$ (MGS17). The Laser dataset is a real-world dataset composed of laser-generated data recorded from a far-infrared laser in a chaotic state. To evaluate the robustness of the proposed model, we created two new datasets by adding Gaussian noise to the above-mentioned two datasets and named them Noisy-MGS17 and Noisy-laser. The glimpses of four datasets are shown in Fig. D.1. The partition of datasets are listed in Table D.1.

We performed 84-step-ahead prediction tasks with MGS17 and Noisy-MGS17 datasets and performed one-step-ahead prediction tasks with Laser and Noisy-Laser datasets. We compared prediction performances of our proposed model with those of the ESN [11], the SPESN [63], the DeepESN [14], the MSLESN [15],

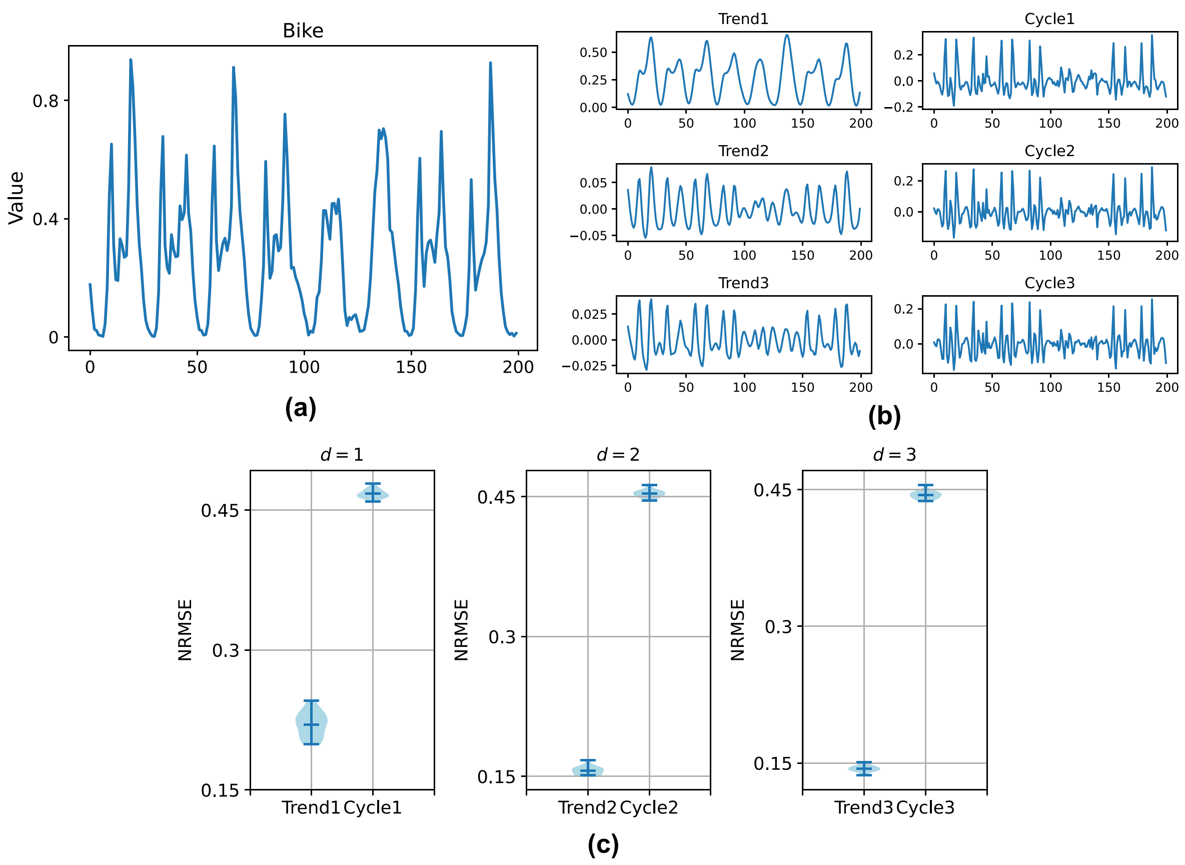

Fig. C.2. (a) The original time series of Bike dataset. (b) Trend and cycle components obtained from the data in (a) through three decompositions with the proposed time series decomposer in the HP-MRESN. (c) A violin plot of prediction performances averaged over 20 trials for the three trend and cycle components in (b).

Table D.1
Data partition for the four chaotic time-series datasets.

	Training	Validation	Testing	Initial transient
MGS17	3000	1000	1000	100
Noisy-MGS17	3000	1000	1000	100
Laser	2000	500	500	100
Noisy-laser	2000	500	500	100

Table D.2
Average NRMSEs ± (std) for the tested models for the four chaotic time-series datasets.

	MGS17	Laser	Noisy-MGS17	Noisy-laser
ESN	3.17E−02 ± (1.04E−02)	4.59E−02 ± (2.18E−03)	4.64E−01 ± (1.98E−02)	8.48E−01 ± (1.00E−03)
SPESN	3.71E−02 ± (1.53E−02)	4.69E−02 ± (1.61E−03)	4.47E−01 ± (1.65E−02)	8.27E−01 ± (2.42E−03)
DeepESN	**1.00E−04** ± **(1.75E−05)**	5.00E−02 ± (2.20E−03)	**4.31E−01** ± **(2.15E−03)**	8.45E−01 ± (5.89E−04)
MSLESN	4.51E−04 ± (4.45E−05)	7.39E−02 ± (8.72E−03)	4.34E−01 ± (2.36E−03)	8.46E−01 ± (1.11E−03)
GroupedESN	8.66E−03 ± (1.34E−03)	4.66E−02 ± (1.95E−03)	4.34E−01 ± (6.14E−04)	8.49E−01 ± (7.77E−04)
HP-MRESN	4.72E−02 ± (8.49E−03)	**2.09E−02** ± **(4.17E−04)**	4.62E−01 ± (5.13E−03)	**1.08E−01** ± **(1.38E−03)**

and the GroupedESN [17]. The SPESN is inspired by the Scale-free Highly-clustered Echo state Network (SHESN) [64], and the optimized network topology of the reservoir is generated by Partitioning Around Medoids (PAM) algorithm [65]. Since Ref. [63] shows that the prediction performances of SPESN are better than those of the ESN and SHESN, we adopted the SPESN as a tested model in the following experiments.

The hyperparameter settings of the ESN, DeepESN, MSLESN, GroupedESN, and HP-MRESN are the same as those described in Section 5.3. We used the scikit-learn-extra library [66] to reproduce the clustered reservoir topology of the SPESN and searched the optimal values of other parameters according to those listed in Table 2.

The best prediction performances of our proposed models and the other tested models for the four chaotic time-series prediction tasks are listed in Table D.2. From this table, we can find that the proposed HP-MRESN outperforms the other tested models for the Laser and Noisy-laser datasets. However, prediction performances

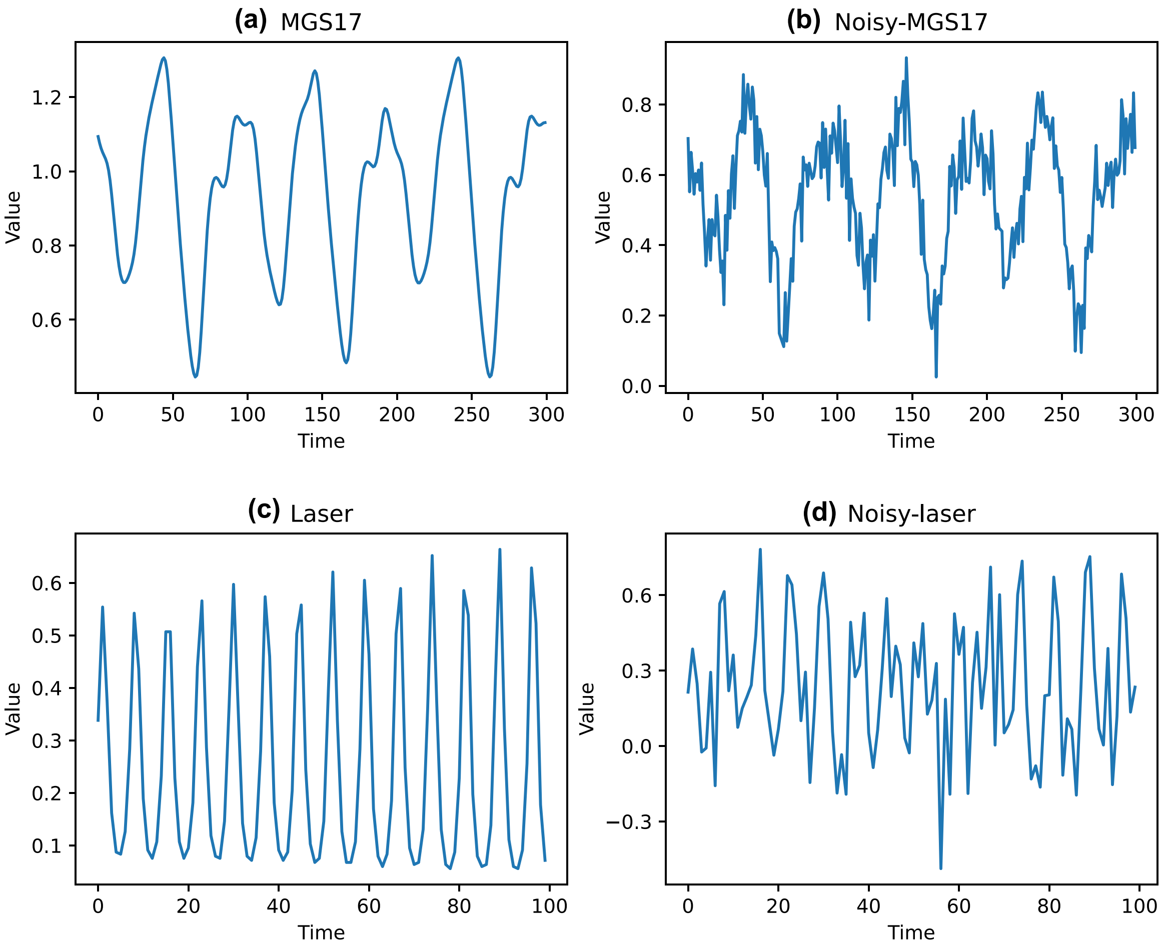

Fig. D.1. Glimpses of four chaotic time-series datasets. (a) MGS17, (b) Noisy-MGS17, (c) Laser, and (d) Noisy-laser. The Noisy-MGS17 dataset was generated by adding Gaussian noise with mean zero and standard deviation 0.1 to the MGS17 dataset [61]. The Noisy-laser dataset was created by adding Gaussian noise with mean zero and standard deviation 0.2 to the Laser dataset [62].

Table D.3
The optimal hyperparameter settings in the HP-MRESN for the four chaotic time-series datasets.

Datasets	Parameters
MGS17	$N_R = 500$, $\alpha = 0.3$, $\delta = 1$, $N_D = 1$
Laser	$N_R = 500$, $\alpha = 0.6$, $\delta = 1$, $N_D = 6$
Noisy-MGS17	$N_R = 500$, $\alpha = 0.1$, $\delta = 1$, $N_D = 8$
Noisy-laser	$N_R = 500$, $\alpha = 0.3$, $\delta = 1$, $N_D = 10$

of the HP-MRESN are worse than those of the other MRESN models for the MGS17 and the Noisy-MGS17 datasets. These results suggest that our proposed model is more effective and robust than the other single and multiple reservoir ESN models when dealing with real-world time-series datasets. We list the best hyperparameter settings of the HP-MRESN for the four chaotic time-series datasets in Table D.3.

References

[1] M. Casdagli, Nonlinear prediction of chaotic time series, Physica D (ISSN: 0167-2789) 35 (3) (1989) 335–356.
[2] W. Xu, H. Peng, X. Zeng, F. Zhou, X. Tian, X. Peng, Deep belief network-based AR model for nonlinear time series forecasting, Appl. Soft Comput. 77 (2019) 605–621.
[3] B. Lim, S. Zohren, Time-series forecasting with deep learning: A survey, Phil. Trans. R. Soc. A 379 (2194) (2021) 20200209.
[4] J.L. Elman, Finding structure in time, Cogn. Sci. 14 (2) (1990) 179–211.
[5] S. Li, W. Li, C. Cook, C. Zhu, Y. Gao, Independently recurrent neural network (indrnn): Building a longer and deeper RNN, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 5457–5466.
[6] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. (ISSN: 0899-7667) 9 (8) (1997) 1735–1780, http://dx.doi.org/10.1162/neco.1997.9.8.1735.
[7] K. Cho, B. Van Merriënboer, D. Bahdanau, Y. Bengio, On the properties of neural machine translation: Encoder-decoder approaches, 2014, arXiv preprint arXiv:1409.1259.
[8] P.J. Werbos, Backpropagation through time: What it does and how to do it, Proc. IEEE 78 (10) (1990) 1550–1560.
[9] M.P. Cuéllar, M. Delgado, M.d.C.P. Jiménez, An application of non-linear programming to train recurrent neural networks in time series prediction problems, in: ICEIS (2), 2005, pp. 35–42.
[10] H. Jaeger, The "Echo State" Approach to Analyzing and Training Recurrent Neural Networks-with An Erratum Note, Vol. 148, German National Research Center for Information Technology GMD Technical Report, Bonn, Germany, 2001, p. 13, 34.
[11] M. Lukoševičius, H. Jaeger, Reservoir computing approaches to recurrent neural network training, Comp. Sci. Rev. 3 (3) (2009) 127–149.
[12] G. Tanaka, T. Yamane, J.B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, A. Hirose, Recent advances in physical reservoir computing: A review, Neural Netw. 115 (2019) 100–123.
[13] C. Sun, M. Song, S. Hong, H. Li, A review of designs and applications of echo state networks, 2020, arXiv preprint arXiv:2012.02974.
[14] C. Gallicchio, A. Micheli, L. Pedrelli, Design of deep echo state networks, Neural Netw. 108 (2018) 33–47.
[15] T. Akiyama, G. Tanaka, Computational efficiency of multi-step learning echo state networks for nonlinear time series prediction, IEEE Access 10 (2022) 28535–28544.
[16] F. Triefenbach, A. Jalalvand, B. Schrauwen, J.-P. Martens, Phoneme recognition with large hierarchical reservoirs, Adv. Neural Inf. Process. Syst. 23 (2010) 2307–2315.
[17] C. Gallicchio, A. Micheli, L. Pedrelli, Deep reservoir computing: A critical experimental analysis, Neurocomputing 268 (2017) 87–99.

[18] C. Gallicchio, A. Micheli, Richness of deep echo state network dynamics, in: International Work-Conference on Artificial Neural Networks, Springer, 2019, pp. 480–491.
[19] Z. Li, G. Tanaka, Deep echo state networks with multi-span features for nonlinear time series prediction, in: 2020 International Joint Conference on Neural Networks, IJCNN, IEEE, 2020, pp. 1–9.
[20] T. Akiyama, G. Tanaka, Analysis on characteristics of multi-step learning echo state networks for nonlinear time series prediction, in: 2019 International Joint Conference on Neural Networks, IJCNN, IEEE, 2019, pp. 1–8.
[21] Z. Tong, G. Tanaka, Reservoir computing with untrained convolutional neural networks for image recognition, in: 2018 24th International Conference on Pattern Recognition, ICPR, IEEE, 2018, pp. 1289–1294.
[22] A. Deihimi, O. Orang, H. Showkati, Short-term electric load and temperature forecasting using wavelet echo state networks with neural reconstruction, Energy 57 (2013) 382–401.
[23] X. Yao, Z. Wang, Broad echo state network for multivariate time series prediction, J. Franklin Inst. B 356 (9) (2019) 4888–4906.
[24] X. Sun, T. Li, Q. Li, Y. Huang, Y. Li, Deep belief echo-state network and its application to time series prediction, Knowl.-Based Syst. 130 (2017) 17–29.
[25] X. Xu, W. Ren, A combination model based on EEMD-PE and echo state network for chaotic time series prediction, in: 2019 Eleventh International Conference on Advanced Computational Intelligence, ICACI, IEEE, 2019, pp. 290–295.
[26] I. Daubechies, Orthonormal bases of compactly supported wavelets, in: Fundamental Papers in Wavelet Theory, Princeton University Press, 2009, pp. 564–652.
[27] N. Amjady, F. Keynia, Short-term load forecasting of power systems by combination of wavelet transform and neuro-evolutionary algorithm, Energy 34 (1) (2009) 46–57.
[28] Z. Wu, N.E. Huang, Ensemble empirical mode decomposition: A noise-assisted data analysis method, Adv. Adapt. Data Anal. 1 (01) (2009) 1–41.
[29] A. Savitzky, M.J. Golay, Smoothing and differentiation of data by simplified least squares procedures, Anal. Chem. 36 (8) (1964) 1627–1639.
[30] R.J. Hodrick, E.C. Prescott, Postwar US business cycles: An empirical investigation, J. Money, Credit, Bank. (1997) 1–16.
[31] J.D. Hamilton, Why you should never use the hodrick-prescott filter, Rev. Econ. Stat. 100 (5) (2018) 831–843.
[32] Z. Li, G. Tanaka, Multi-reservoir echo state networks with sequence resampling for nonlinear time-series prediction, Neurocomputing 467 (2022) 115–129.
[33] 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.
[34] H. Jaeger, Tutorial on Training Recurrent Neural Networks, Covering BPPT, RTRL, EKF and the "Echo State Network" Approach, Vol. 5, GMD-Forschungszentrum Informationstechnik Bonn, 2002.
[35] D.W. Marquardt, R.D. Snee, Ridge regression in practice, Amer. Statist. 29 (1) (1975) 3–20.
[36] A.N. Tikhonov, A. Goncharsky, V. Stepanov, A.G. Yagola, Numerical Methods for the Solution of Ill-Posed Problems, Vol. 328, Springer Science & Business Media, 2013.
[37] H. Abdi, L.J. Williams, Principal component analysis, Wiley Interdiscip. Rev. Comput. Stat. 2 (4) (2010) 433–459.
[38] P. Chen, R. Liu, K. Aihara, L. Chen, Autoreservoir computing for multistep ahead prediction based on the spatiotemporal information transformation, Nature Commun. 11 (1) (2020) 1–15.
[39] S.W.D. Center, The international sunspot number, international sunspot number monthly bulletin and online catalogue, 1749-2019.
[40] Z. Li, G. Tanaka, A multi-reservoir echo state network with multiple-size input time slices for nonlinear time-series prediction, in: International Conference on Neural Information Processing, Springer, 2021, pp. 28–39.
[41] Open power system data, 2006-2019, URL https://data.open-power-system-data.org/.
[42] H. Fanaee-T, J. Gama, Event labeling combining ensemble detectors and background knowledge, Prog. Artif. Intell. (ISSN: 2192-6352) (2013) 1–15, http://dx.doi.org/10.1007/s13748-013-0040-3.
[43] X. Chen, Z. He, J. Wang, Spatial-temporal traffic speed patterns discovery and incomplete data recovery via SVD-combined tensor decomposition, Transp. Res. C 86 (2018) 59–77.
[44] G.E. Hinton, A practical guide to training restricted Boltzmann machines, in: Neural Networks: Tricks of the Trade, Springer, 2012, pp. 599–619.
[45] N.E. Huang, Z. Shen, S.R. Long, M.C. Wu, H.H. Shih, Q. Zheng, N.-C. Yen, C.C. Tung, H.H. Liu, The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis, Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci. 454 (1971) (1998) 903–995.
[46] E.R. Girden, ANOVA: Repeated Measures, no. 84, SAGE Publications, Inc., 1992.
[47] J.W. Tukey, Comparing individual means in the analysis of variance, Biometrics (1949) 99–114.
[48] S. Herbold, Autorank: A python package for automated ranking of classifiers, J. Open Source Softw. 5 (48) (2020) 2173, http://dx.doi.org/10.21105/joss.02173.
[49] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, Pytorch: An imperative style, high-performance deep learning library, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems 32, Curran Associates, Inc., 2019, pp. 8024–8035.
[50] S. Seabold, J. Perktold, Statsmodels: Econometric and statistical modeling with python, in: Proceedings of the 9th Python in Science Conference, Vol. 57, Austin, TX, 2010, pp. 10–25080.
[51] D. Laszuk, Python implementation of empirical mode decomposition algorithm, GitHub Repos. (2017) http://dx.doi.org/10.5281/zenodo.5459184.
[52] P. Virtanen, R. Gommers, T.E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S.J. van der Walt, M. Brett, J. Wilson, K.J. Millman, N. Mayorov, A.R.J. Nelson, E. Jones, R. Kern, E. Larson, C.J. Carey, İ. Polat, Y. Feng, E.W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E.A. Quintero, C.R. Harris, A.M. Archibald, A.H. Ribeiro, F. Pedregosa, P. van Mulbregt, Scipy 1.0 contributors, scipy 1.0: Fundamental algorithms for scientific computing in python, Nature Methods 17 (2020) 261–272, http://dx.doi.org/10.1038/s41592-019-0686-2.
[53] G. Lee, R. Gommers, F. Waselewski, K. Wohlfahrt, A. O'Leary, Pywavelets: A python package for wavelet analysis, J. Open Source Softw. 4 (36) (2019) 1237.
[54] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2015, CoRR abs/1412.6980.
[55] Z. Li, G. Tanaka, HP-ESN: Echo state networks combined with hodrick-prescott filter for nonlinear time-series prediction, in: 2020 International Joint Conference on Neural Networks, IJCNN, IEEE, 2020, pp. 1–9.
[56] T. Kim, B.R. King, Time series prediction using deep echo state networks, Neural Comput. Appl. 32 (23) (2020) 17769–17787.
[57] M.R. Bonyadi, Z. Michalewicz, Particle swarm optimization for single objective continuous space problems: A review, Evol. Comput. 25 (1) (2017) 1–54.
[58] P.I. Frazier, A tutorial on Bayesian optimization, 2018, arXiv preprint arXiv:1807.02811.
[59] H. Zhou, Y. Zhang, W. Duan, H. Zhao, Nonlinear systems modelling based on self-organizing fuzzy neural network with hierarchical pruning scheme, Appl. Soft Comput. 95 (2020) 106516.
[60] H. Zhou, Y. Li, Q. Zhang, H. Xu, Y. Su, Soft-sensing of effluent total phosphorus using adaptive recurrent fuzzy neural network with gustafson-kessel clustering, Expert Syst. Appl. (2022) 117589.
[61] M.C. Mackey, L. Glass, Oscillation and chaos in physiological control systems, Science 197 (4300) (1977) 287–289.
[62] A.S. Weigend, N.A. Gershenfeld, Results of the time series prediction competition at the santa fe institute, in: IEEE International Conference on Neural Networks, IEEE, 1993, pp. 1786–1793.
[63] E. Najibi, H. Rostami, SCESN, SPESN, SWESN: Three recurrent neural echo state networks with clustered reservoirs for prediction of nonlinear and chaotic time series, Appl. Intell. 43 (2) (2015) 460–472.
[64] Z. Deng, Y. Zhang, Collective behavior of a small-world recurrent neural system with scale-free distribution, IEEE Trans. Neural Netw. 18 (5) (2007) 1364–1375.
[65] L. Kaufmann, Clustering by means of medoids, in: Proc. Statistical Data Analysis Based on the L1 Norm Conference, Neuchatel, 1987, 1987, pp. 405–416.
[66] Scikit-learn-extra, 2021, URL https://scikit-learn-extra.readthedocs.io/en/stable/.