Improving Deep Echo State Network with Neuronal Similarity-based Iterative Pruning Merging Algorithm

Qingyu Shen , Hanwen Zhang , Yao Mao *

Posted Date: 30 January 2023

doi: 10.20944/preprints202301.0533.v1

Keywords: reservoir computing; deep echo state network; neuronal similarity-based iterative pruning merging algorithm; chaotic time series forecast

Article

Improving Deep Echo State Network with Neuronal Similarity-Based Iterative Pruning Merging Algorithm

QingyuShen [1,2,3], HanwenZhang [1,2,3] and Yao Mao [1,2,3,*]

[1] Key Laboratory of Optical Engineering, Chinese Academy of Sciences, Chengdu 610209, China
[2] Institute of Optics and Electronics, Chinese Academy of Sciences, Chengdu 610209, China
[3] University of Chinese Academy of Sciences, Beijing 1000049, China
* Correspondence: maoyao@ioe.ac.cn

Abstract: Recently, a layer-stacked ESN model named deep echo state network (DeepESN) has been established. As an interactional model of recurrent neural network and deep neural network, investigations of DeepESN are of significant importance in both areas. Optimizing the structure of neural networks remains a common task in artificial neural networks, and the question of how many neurons should be used in each layer of DeepESN must be stressed. In this paper, our aim is to solve the problem of choosing the optimized size of DeepESN. Inspired by the sensitive iterative pruning algorithm, a neuronal similarity-based iterative pruning merging algorithm (NS-IPMA) is proposed to iteratively prune or merge the most similar neurons in DeepESN. Two chaotic time series prediction tasks are applied to demonstrate the effectiveness of NS-IPMA. The results show that the DeepESN pruned by NS-IPMA outperforms the unpruned DeepESN with the same network size, and NS-IPMA is a feasible and superior approach to improving the generalization performance of DeepESN.

Keywords: reservoir computing; deep echo state network; neuronal similarity-based iterative pruning merging algorithm; chaotic time series forecast

1. Introduction

Recurrent neural networks (RNNs) represent a consolidated computational abstraction for learning with variable length time series data [1]. As a simplified paradigm of RNN, the echo state network [2,3] (ESN) provides a prominent reduction in the computational cost compared to other paradigms of RNNs(e.g. ([4], LSTM), ([5], GRU)), for which the hidden layer of ESN is constructed by a randomly generated reservoir instead of independent neurons and the output weights of ESN are trained by a simple linear regression rather than backpropagation algorithm. Thus, ESN has a successful application in various time series prediction problems (e.g. [6–9]).

Deep neural networks [10] (DNNs) have the potential to learn data representations at various levels of abstraction and are being increasingly stressed in the machine learning community. Recently, a layer-stacked ESN model named deep echo state network (DeepESN) has been established and investigated, theoretically and experimentally, by Gallicchio, etc. The inherent characterization of the system dynamics developed at the different layers of DeepESN is experimentally analyzed in [11] and theoretically explained in[12]; A theoretical foundation for the study of DeepESN from a dynamical system point of view is introduced in [13]; Further details on the analysis and advancements of DeepESN could be found in [14]. As an interactional model of RNN and DNN, investigations of DeepESN are of significant importance in both areas. On the one hand, DeepESN expands our knowledge on how the information with memory attracted by RNN is extracted by hierarchical neural networks, on the other hand, DeepESN helps us better understand how the abstract intrinsic representations of time series extracted by DNN are recalled in reservoirs. Furthermore, DeepESN has richer nonlinear representation capacity and less computational complexity, and better predictive performance than single layer ESN [15].

 doi:10.20944/preprints202301.0533.v1

Optimizing the structure of neural networks remains a common task in artificial neural networks, and the same question of how many neurons should be used in each layer must be stressed in all types of neutral network. If the neurons are too few, the architecture does not satisfy the error demand by learning from the data, whereas if the neurons are too many, learning leads to the well-known overfitting problem [16]. As far as we know, little research has been carried out on optimizing architecture of DeepESN. For research that has already been carried out on DeepESN, the same number is commonly assigned in each layer, which is acceptable but not optimal. In this paper, our aim is to solve the problem of choosing the optimized size of DeepESN, especially the number of neuron in different layers.

In 2014, a sensitive iterative pruning algorithm (SIPA) was proposed by Wang and Yan [17] to optimize the simple cycle reservoir network (SCRN), the algorithm was used to prune the least sensitive neurons one by one according to the sensitive analysis, the results showed that the SIPA method can optimize the structure and improve the generalization performance of the SCRN, meanwhile, pruning out redundant neurons could contribute to reducing the calculation and improving the computing efficiency of the network. Inspired by these advantages of SIPA, we wanted to apply a similar iterative pruning approach on DeepESN. However, SCRN is a kind of minimal-complexity ESN with simple cycle topology, the topology of DeepESN is much more complex than SCRN. Pruning a neuron in the network will raise perturbations on adjacent neurons, resulting in unstable network performance, in SIPA, the perturbations could be eliminated by adjusting the input weights into the perturbed neurons to minimum the distance of its input signal before and after pruning. Due to the hierarchical structure of reservoirs in DeepESN, perturbation raised by pruning a neuron in the lower layer will propagate into higher layers layer by layer, leading to greater instability of network performance and difficulty of perturbation elimination.

In order to overcome above difficulty, a new neuronal similarity-based iterative pruning merging algorithm (NS-IPMA) is proposed to iteratively prune out or merge the most similar neurons in DeepESN. In NS-IPMA, a pair of most similar twin neurons, which is regarded as redundant neurons in the network, are selected out iteratively, then, if they exist in different layers, the one in higher layer will be pruned out, if they are in a same resverior, they will be merged into one neuron, which works as the substitution of antecedent twin neurons. Quantive estimation of neuronal similarity plays an essential role in determining the redundant neurons which should be pruned out, Four neuronal similarity estimation criteria of NS-IPMA approach were attempted, including the inverse of Euclidean distance, Pearson's correlation, Spearman's correlation and Kendall's correlation. Reducing the network size is a directly effective approach to improve generalization performance of neural network, because pruning out neurons will lead to reduction of network size, to verify the effectiveness of NS-IPMA method. The The DeepESNs pruned by the NS-IPMA method were compared with unpruned DeepESNs, whose number of neurons in each reservoir is the same. The pruned DeepESN and the unpruned DeepESN were compared with equal layer number advancements equal total neruon number.The results of the experiment on two chaotic time series prediction tasks showed that the NS-IPMA method has good network structure adaptability , and the DeepESNs pruned by NS-IPMA methed have better generalization performance and better robustness than the unpruned DeepESNs, indicating that the NS-IPMA methed is a feasible and superior approach to improving the generalization performance of DeepESN. NS-IPMA method provides a novel approach for choosing the appropriate network size of DeepESN, it also has application potential in other RNNs and DNNs.

This paper is organized as follows. DeepESN and entropy quantification of reservoir richness is detailly introduced in Section 2, SIPA and new proposed NS-IPMA are described in Section 3, experiments and results are presented and discussed in Section 4. Section 5 draws the conclusions.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 January 2023 doi:10.20944/preprints202301.0533.v1

2. Deep echo state network

2.1. Leakey integrator echo state network

A leakey integrator echo state network [18],(LI-ESN), as shown in Figure 1, is a recurrent neural network with three layers: input layer $\mathbf{u}(t) \in \mathbb{R}^{N_u \times 1}$, hidden layer $\mathbf{x}(t) \in \mathbb{R}^{N_x \times 1}$, output layer $\mathbf{y}(t) \in \mathbb{R}^{N_y \times 1}$. t notes time sequence order. The hidden layer is regarded as a reservoir, which holds the memory of foregone information, and $\mathbf{x}(t)$ is refreshed by state transition function:

$$\mathbf{x}(t) = \alpha \mathbf{x}(t-1) + (1-\alpha)\tanh(\mathbf{W}_i \mathbf{u}(t-1) + \mathbf{W}_r \mathbf{x}(t-1)), \tag{1}$$

where $\mathbf{W}_i \in \mathbb{R}^{N_x \times N_u}$is the input weight matrix randomly generated before training, $\mathbf{W}_r \in \mathbb{R}^{N_x \times N_x}$ is the reservoir weight matrix previously given before training. $\alpha \in [0,1]$ is the leaky parameter. $\tanh(\bullet)$ is the activation function of the hidden layer. The reservoir weights in $\mathbf{W}_r$ must be initialized to satisfy the echo state property (ESP) [19,20], denoting by $\rho(\bullet)$ the spectral radius operator (*i.e.* the largest absolute eigenvalue of its matrix argument), the necessary condition for the ESP is expressed as follows:

$$\rho\left((1-\alpha)\mathbf{I} + \alpha \mathbf{W}_r\right) < 1. \tag{2}$$

Accordingly,the values in matrix $\mathbf{W}_r$ are randomly selected from a uniform distribution(e.g.$U[-1,1]$), and then rescaled to satisfy above condition in equation 2.

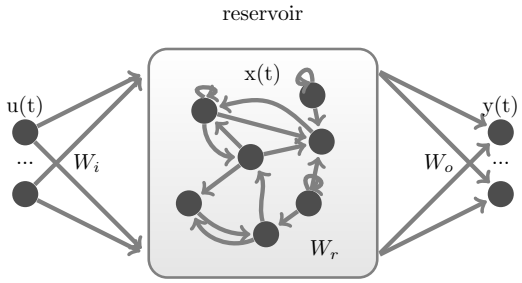


Figure 1. Structure of LI-ESN.

The output $\mathbf{y}(t)$ can be calculated through a linear combination of reservoir states as follows:

$$\mathbf{y}(t) = \mathbf{W}_o \mathbf{x}(t), \tag{3}$$

where $\mathbf{W}_o \in \mathbb{R}^{N_y \times N_x}$is the output weight matrix.

During training, the states of reservoir neurons are collected in a training state matrix $\mathbf{X}_{train} = [\mathbf{x}(1), \mathbf{x}(2) \ldots \mathbf{x}(L_{train})]$, and an output target matrix $\mathbf{Y}_{train} = [\hat{\mathbf{y}}(1), \hat{\mathbf{y}}(2) \ldots \hat{\mathbf{y}}(L_{train})]$ is collected correspondingly, where L_{train} is the number of training samples.

The output weights in $\mathbf{W}_o$ can be calculated by ridge regression as follows:

$$\mathbf{W}_o = \left((\mathbf{X}_{train}^T \mathbf{X}_{train} + \lambda \mathbf{I})^{-1} \mathbf{X}_{train}^T \mathbf{Y}_{train}\right)^T, \tag{4}$$

where $(\bullet)^T$ represents matrix transpose , $(\bullet)^{-1}$ represents matrix inversion, $\lambda \mathbf{I}$ is a regularization term ensuring $\mathbf{X}_{train}^T \mathbf{X}_{train}$ is invertible.

 doi:10.20944/preprints202301.0533.v1

2.2. Deep echo state network

DeepESN was first introduced by Gallicchio [11,14], as a stacked reservoir computing (RC) architecture, multiple reservoir layers are stacked one on top of each other. The state transition functions of hidden layers in DeepESN are expressed as:

$$\begin{cases} \mathbf{x}^1(t) = \alpha^1\mathbf{x}^1(t) + (1-\alpha^1)\tanh(\mathbf{W}_i\mathbf{u}(t-1) + \mathbf{W}_r^1\mathbf{x}^1(t-1)), & l = 1 \\ \mathbf{x}^l(t) = \alpha^l\mathbf{x}^l(t) + (1-\alpha^l)\tanh(\mathbf{W}_p^{l-1}\mathbf{x}^{l-1}(t-1) + \mathbf{W}_r^l\mathbf{x}^l(t-1)), & l \in [2, L] \end{cases} \tag{5}$$

where the superscript(1 and l) is the layer notation,with totally L hidden layers in the network. $\mathbf{x}^l(t) \in \mathbb{R}^{N_x^l \times 1}$ represents the l-th hidden layer (i.e. reservoir(l)) with N_x^l neurons inside, $\mathbf{W}_i \in \mathbb{R}^{N_x^1 \times N_u}$ is the input weight matrix of the first hidden layer, $\mathbf{W}_r^l \in \mathbb{R}^{N_x^l \times N_x^l}$ is the reservoir weight matrix of the l-th hidden layer, $\mathbf{W}_p^{l-1} \in \mathbb{R}^{N_x^{l-1} \times N_x^l}$ is the propagate weight matrix which connects reservoir$(l-1)$ to reservoir(l).

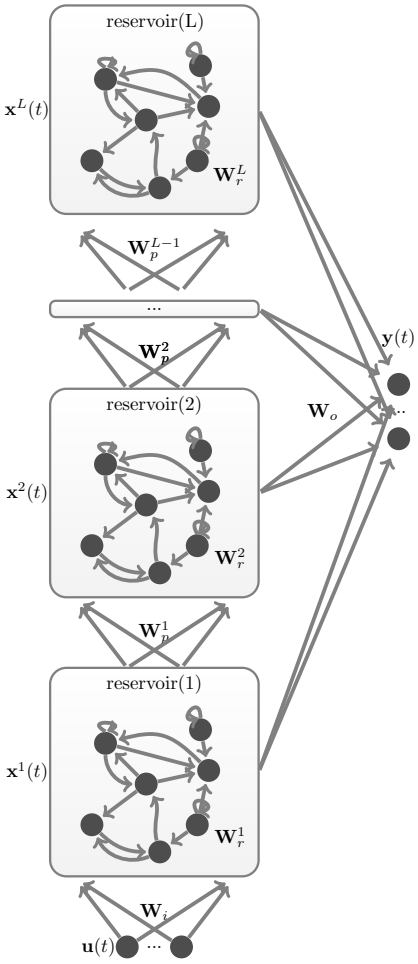


Figure 2. Structure of DeepESN.

As in the standard LI-ESN approach, the reservoir weights of a DeepESN are initialized subject to similar stability constraints. In the case of DeepESN, such constraints are expressed by the necessary condition for the ESP of deep RC networks [13], described by the following equation:

$$\max_{l \in [1,L]} \rho\left((1-\alpha^l)\mathbf{I} + \alpha^l\mathbf{W}_r\right) < 1, \tag{6}$$

a same leaky parameter ($\alpha^l \equiv \alpha$) in each layer is considered in this paper.

 doi:10.20944/preprints202301.0533.v1

The values in each reservoir matrix$\{\mathbf{W}_r^l | l \in [1, L]\}$ are randomly initialized from a uniform distribution (e.g.$U[-1, 1]$), after that, each $\mathbf{W}_r^l$ is spectral normalized by its spectral radius and rescaled by a same reservoir scaling paramerter γ_r to meet the demand of equation 6.

$$\mathbf{W}_r^l \leftarrow \frac{\gamma_r \mathbf{W}_r^l}{\rho(\mathbf{W}_r^l)}, \quad l \in [1, L]. \tag{7}$$

The values in input weight matrix $\mathbf{W}_i$ are randomly selected from a uniform distribution $U[-\gamma_i, \gamma_i]$, where γ_i is the input scaling parameter. The values in each propagate weight matrix $\{\mathbf{W}_p^l | l \in [1, L-1]\}$ are randomly selected from a uniform distribution $U[-\gamma_p, \gamma_p]$ where γ_p is the propagate scaling parameter.

The output equation and the training equation of DeepESN are formed by concatenating all hidden neurons in each reservoirs together, denoting $\tilde{\mathbf{x}}(t)^T = [\mathbf{x}^1(t)^T \ \mathbf{x}^2(t)^T \dots \mathbf{x}^L(t)^T]$ and substituting $\tilde{\mathbf{x}}(t)$ for $\mathbf{x}(t)$ in equation 3 and equation 4:

$$\mathbf{y}(t) = \mathbf{W}_o \tilde{\mathbf{x}}(t), \tag{8}$$

$$\mathbf{W}_o = \left((\tilde{\mathbf{X}}_{train}^T \tilde{\mathbf{X}}_{train} + \lambda \mathbf{I})^{-1} \tilde{\mathbf{X}}_{train}^T \mathbf{Y}_{train} \right)^T. \tag{9}$$

2.3. Architechiecural richness of DeepESN

The components of the state should be as diverse as possible to provide a richer pool of dynamics from which the trainable part can appropriately combine. From an information-theoretic point of view, this form of richness could be measured by means of the entropy of instantaneous reservoir states [1]. Here an efficient estimator of Renyi's quadratic entropy is introduced: suppose that we have N independent and identically distributed samples $\{v_1, \dots, v_N\}$ for the continuous random variable $\mathbf{V}$, An estimation of Renyi entropy directly from sampling data is defined as:

$$H_2(\mathbf{V}) = -\log\left(\frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} G_{\kappa\sqrt{2}}(v_j - v_i)\right), \tag{10}$$

where $G_{\kappa\sqrt{2}}(\bullet)$ is a Gauss kernel function with standard deviation $\kappa\sqrt{2}$, κ could be determined by Silverman's rule:

$$\kappa = \sigma(\mathbf{V})(4N^{-1}(2d+1)^{-1})^{\frac{1}{d+4}}, \tag{11}$$

where $\sigma(\mathbf{V})$ is standard deviation and d is the data dimensionality.

Average state entropy (ASE) is obtained by time average of instantaneous Renyi's quadratic estimation of reservoir neurons.

$$\mathcal{H}(t) = H_2(\tilde{\mathbf{x}}(t)), \tag{12}$$

$$ASE = \frac{1}{S} \sum_{t=1}^{S} \mathcal{H}(t), \tag{13}$$

where S is the sample number. ASE gives us a research perspective independent of learning aspect, higher ASE values are preferable and denote richer dynamics in reservoirs[1].

3. Pruning deep echo state network with neuronal similarity-based iterative pruning merging algorithm

3.1. Sensitive iterative pruning algorithm on simple cycle reservoir network

The simple cycle reservoir network(SCRN)is a kind of minimum complexity ESN, which has a cycle topology in the reservoir [21]. Every reservoir neuron is unidirectionally connected to its two

 doi:10.20944/preprints202301.0533.v1

adjacent neurons, A SIPA method is introduced to choose the right network size of SCRN through iteratively pruning out the least sensitive neurons [17].

SIPA method is carried out in the following steps:

Step 1. Establish an SCRN with a large enough reservoir and satisfactory performance.

Step 2. Select a neuron to be pruned using the sensitive criterion [22] (assume x_m is to be pruned), as diagramed in Figure 3, remove all weights connected to x_m as follows:

$$\begin{cases} \mathbf{W}_i(m,:) \leftarrow \mathbf{0} \\ \mathbf{W}_r(m,m-1) \leftarrow 0 \\ \mathbf{W}_r(m+1,m) \leftarrow 0 \end{cases} \tag{14}$$

Step 3. Establish a new link between two neighbors of pruned neuron, the link weight is determined to eliminate the perturbation caused by pruning, denating the input to x_{m+1} before pruning $I_o = \mathbf{W}_i(m+1,:)\mathbf{u} + \mathbf{W}_r(m+1,m)x_m$, the input to x_{m+1} after pruning $I_n = \mathbf{W}_i(m+1,:)\mathbf{u} + \mathbf{W}_r(m+1,m-1)x_{m-1}$, the perturbation is eliminated and the original reservoir behavior is maintained as long as I_n is set as close as possible to I_o, by solving the following optimization problem:

$$\min_{\mathbf{W}_r(m+1,m-1)} ||I_o - I_n||_2. \tag{15}$$

Step 4. Adjust the output weights by retraining the network.Then calculate the training error.

Step 5. Repeat steps 2-4 until the training error or the reservoir size reaches an acceptable range.

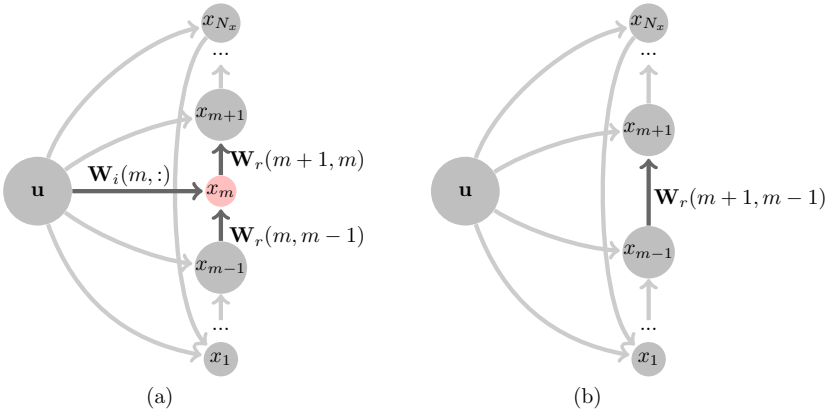


Figure 3. Connection weight coefficients diagram of neurons in reservoir of SCRN before(a) and after(b) pruning a neuron by SIPA.

The key to a successful application of SIPA is Step 3. In Step 2, a reservoir perturbation is triggered by pruning, making the performance of the pruned network unpredictable. Thus, the essential task of Step 3 is to reduce the effects of perturbation so that the rest neurons remain unchanged and the network performance is approximately as good as before.

3.2. Neuronal similarity-based iterative pruning merging algorithm on deep echo state network

Due to the simple topology of SCRN, only one neuron (x_{m+1}) receives input from the pruned neuron (x_m). The perturbation elimination in SCRN is easy to perform. However, the topology of DeepESN is much more complicated, more extensive perturbation will be raised by pruning one neuron in DeepESN because, in a highly coupled reservoir, perturbation at any neuron will be diffused to every neuron of same reservoir. In addition, in hierarchically stacked reservoirs, perturbation in lower layer will be transmitted to every higher layer above. It is very difficult to eliminate all of these perturbations.

 doi:10.20944/preprints202301.0533.v1

The merging method is designed to solve this difficulty, this coincides with Islam's idea [23], two neurons are merged by averaging their input weights. Consider the following ideal scenario: Merging two identical twin neurons in a same reservoir will derive a new neuron that is identical to anteceden two twins, this new burned neuron could act as eqivalent substitution for anteceden two twins. Conseqently, a neuron is pruned without leading any perturbation through superimposing the output weights of the merged twins as well. This ideal perturbation-free merging has the prerequisite that two identical neurons to be found in the same reservoir, the more similar the two merged neurons are, the weaker perturbation will be raised by merging. Neuronal similarity could be assessed by some quantitative relations of the collected training state matrix. Distance and correlation are commonly used to quantify similarity, four similarity estimation criteria are given in this paper including the inverse of Euclidean distance (ED), Pearson's correlation coefficient (PC),Spearman's correlation coefficient (SC) and Kendall's correlation coefficient (KC) as follows:

Noting the total number of neurons in all reservoirs $M = \sum_{l=0}^{L} N_x^l$, renote the train state matrix

$$\tilde{\mathbf{X}}_{train} = \begin{bmatrix} \mathbf{x}^1(1) & \mathbf{x}^1(2) & \dots & \mathbf{x}^1(L_{train}) \\ \mathbf{x}^2(1) & \mathbf{x}^2(2) & \dots & \mathbf{x}^1(L_{train}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}^L(1) & \mathbf{x}^L(2) & \dots & \mathbf{x}^L(L_{train}) \end{bmatrix}_{(M \times L_{train})} = \begin{bmatrix} \mathbf{n}_1 \\ \mathbf{n}_2 \\ \vdots \\ \mathbf{n}_M \end{bmatrix}, \tag{16}$$

where L_{train} is the number of training samples,$\mathbf{n}_i$ is the historical state of i-th neuron state during training. The similarity of $\mathbf{n}_i$ and $\mathbf{n}_j$ is derived by:

$$ED(i,j) = \frac{1}{\| \mathbf{n}_i - \mathbf{n}_j \|_2}, \tag{17}$$

$$PC(i,j) = \frac{\sigma_{\mathbf{n}_i\mathbf{n}_j}}{\sqrt{\sigma_{\mathbf{n}_i\mathbf{n}_i}\sigma_{\mathbf{n}_j\mathbf{n}_j}}}, \tag{18}$$

$$SC(i,j) = 1 - \frac{6\sum_{t=1}^{L_{train}} d_t^2}{L_{train}\left({L_{train}}^2 - 1\right)}, \tag{19}$$

$$KC(i,j) = \frac{c-d}{\frac{1}{2}L_{train}\left(L_{train}-1\right)}, \tag{20}$$

where $\sigma_{\mathbf{n}_i\mathbf{n}_j}$ represents cross-correlation of $\mathbf{n}_i$ and $\mathbf{n}_j$, $\sigma_{\mathbf{n}_i\mathbf{n}_i}$ and $\sigma_{\mathbf{n}_j\mathbf{n}_j}$ represents autocorrelation of $\mathbf{n}_i$ and $\mathbf{n}_j$. d_t is the rank difference of $\mathbf{n}_i(t)$ and $\mathbf{n}_j(t)$, c is the number of concordant pairs and d is the number of discordant pairs in $\mathbf{n}_i$ and $\mathbf{n}_j$. NS-IPMA based on different similarity estimation criteria are named correspondingly, for instance, ES-IPMA means NS-IPMA based on the inverse of Euclidean distance criterion, etc.

The NS-IPMA method is carried out in the following steps:

Step 1. Initially generate a performable DeepESN with large enough reservoirs by tuning hyperparameters to minimize the average of training and validate error using Particle Swarm Optimization (PSO) algorithm, please refer to A for more detail on hyperparameter tuning. This DeepESN is a primitive network to implement.

Step 2. Washout the reservoirs, activate the reservoirs using train samples to obtain the training state matrix.

Step 3. Quantify the similarity of each two neurons (using one criterion of equation 17 - 20) and select a pair of the most similar neurons.

Step 4. (1). If selected neurons are in the same reservoir (note as x_i^l and x_j^l), merge them. As diagramed in Figure 4, x_s^l is the son neurons merged as the substitute of its parents

 doi:10.20944/preprints202301.0533.v1

x_i^l and x_j^l. l is the reservoir layer where x_i^l and x_j^l exists in. Related weight matrix $\mathbf{W}_p^{l-1}$,$\mathbf{W}_p^l$ and $\mathbf{W}_r^l$ is refreshed as follows[1]:

$$\begin{cases} \mathbf{W}_p^{l-1}(s,:) \leftarrow \dfrac{\mathbf{W}_p^{l-1}(i,:) + \mathbf{W}_p^{l-1}(j,:)}{2} \\ \mathbf{W}_r^l(s,:) \leftarrow \dfrac{\mathbf{W}_r^l(i,:) + \mathbf{W}_r^l(j,:)}{2} \\ \mathbf{W}_r^l(:,s) \leftarrow \mathbf{W}_r^l(:,i) + \mathbf{W}_r^l(:,j) \\ \mathbf{W}_p^l(:,s) \leftarrow \mathbf{W}_p^l(:,i) + \mathbf{W}_p^l(:,j) \end{cases} \tag{21}$$

(2). If selected neurons are in different reservoirs(note as x_i^m and x_j^l), prune one in high layer (assume $m < l$). Related weight matrix $\mathbf{W}_p^{l-1}$,$\mathbf{W}_p^l$ and $\mathbf{W}_r^l$ is refreshed as follows[1]:

$$\begin{cases} \mathbf{W}_p^{l-1}(j,:) \leftarrow \mathbf{0} \\ \mathbf{W}_r^l(j,:) \leftarrow \mathbf{0} \\ \mathbf{W}_r^l(:,j) \leftarrow \mathbf{0} \\ \mathbf{W}_p^l(:,j) \leftarrow \mathbf{0} \end{cases} \tag{22}$$

Step 5. Adjust the output weights by retraining the network.Then estimate the performance of the current network.

Step 6. Repeat steps 2-5 until the training error or the network size reaches an acceptable range.

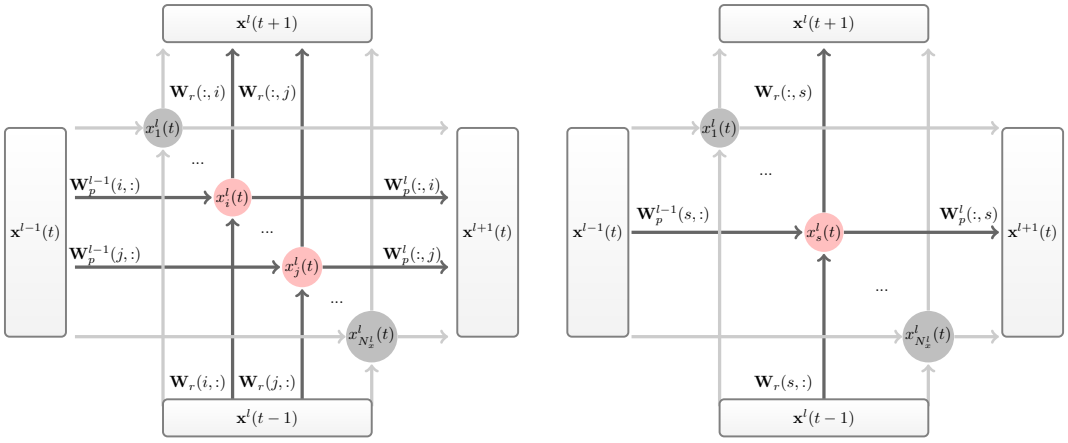


Figure 4. Connection weight coefficients diagram of neurons in reservoir(l) of DeepESN before(a) and after(b) merging a neuron by NS-IPMA.

[1] if $l = 1$, $\mathbf{W}_i$ perform as $\mathbf{W}_p^{l-1}$;if $l = L$,$\mathbf{W}_p^l$ does not exist.

 doi:10.20944/preprints202301.0533.v1

4. Experiments and results

4.1. Datasets

4.1.1. Mackey-Glass chaotic time-seriers

Mackey-Glass time series [24] is a standard benchmark for chaotic time series forecast models where ESN has been successfully applied to demonstrate good performance. The 31 order Mackey-Glass time series is defined in the following differential equation:

$$\frac{dx}{dt} = 0.2\frac{x_{31}}{1 + {x_{31}}^{11}} - 0.1x, \quad x_{31} = x(t - 31), \tag{23}$$

the Mackey Glass dataset (MG) is sampled with a sample frequency of 4Hz. A Python library[25] was used to generate this dataset.

4.1.2. Lorenz chaotic time series

The Lorenz time series prediction [26] is another benchmark problem for ESN. The Lorenz dynamic system is described by the following equations:

$$\begin{cases} \frac{dx}{dt} = -\frac{34}{3}(x - y); \\ \frac{dy}{dt} = -\frac{298}{11}(x - y) + xz; \\ \frac{dz}{dt} = -\frac{17}{7}z + xy \end{cases} \tag{24}$$

The initial values are set as $[x(0), y(0), z(0)] = [0.5, 0, -0.5]$,the Lorenz z axis dataset (LZ) is sampled with a sample frequency of 12.5Hz.

Before the experiments, both datasets are shifted by its mean to remove the DC bias.

4.2. Experiments

4.2.1. Next spot prediction task

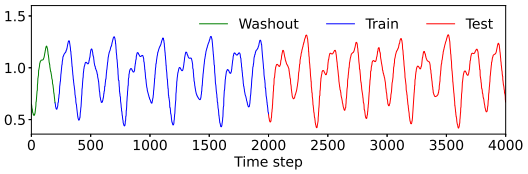
In the next spot prediction experiment, the last four continuous spots were considered the input and the next spot was considered the desired output, i.e.$\{u(t-3), u(t-2), u(t-1), u(t)\}$ was used to predict $u(t+1)$. The number of the train samples and the test samples was 1800 and 2000, 200 samples before the first training sample were used to wash out the initial transient(see Figure 5). The reservoir diversity was quantized by the ASE(see: 2.3) of combined training and testing neuron states activated by training and testing samples. the predictive error performance was quantized by normalized root mean square error:

$$NRMSE = \sqrt{\sum_{t=1}^{T} \frac{\mid \mathbf{y}(t) - \hat{\mathbf{y}}(t) \mid^2}{T \mid \mathbf{y}(t) - \bar{\mathbf{y}}(t) \mid^2}}, \tag{25}$$

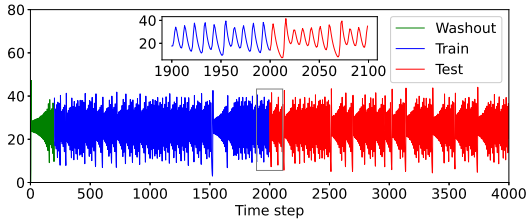
where T is sample number,$\hat{\mathbf{y}}(t)$ is desired output,$\mathbf{y}(t)$ is the readout output and $\bar{\mathbf{y}}(t)$ is average of $\mathbf{y}(t)$.

Two different initial reservoir size conditions of DeepESN were performed in both dataset, the first is 4 stacked reservoir with 100 neurons in each reservoir (Abbreviated to:4×100) and the second is 8 stacked reservoir with 50 neurons in each reservoir (Abbreviated to: 8×50).All experiments were carried out under two different initial conditions on two datasets, model hyperparameters tuned by PSO were recorded in Table A1.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 January 2023 doi:10.20944/preprints202301.0533.v1



(a) *MG*



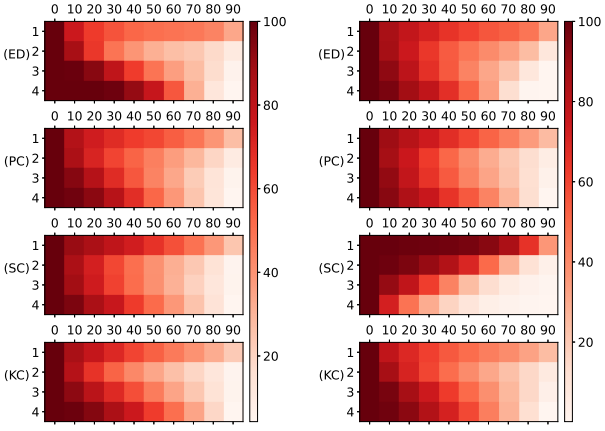
(b) *LZ*

Figure 5. (a) *MG* and (b) *LZ* datasets.

4.2.2. Ablation experiment and control experiment

In order to demonstrate the effectiveness of different similarity estimation criteria, the worst-case scenario of NS-IPMA, Neuron iterative pruning merging algorithm without similarity estimation (IPMA), was investigated as an ablation experiment. in IPMA, the similarity of each two neurons was assigned by random values. Thus, two random neurons would be recognized as the most similar neuron pairs and would be pruned (or merged).

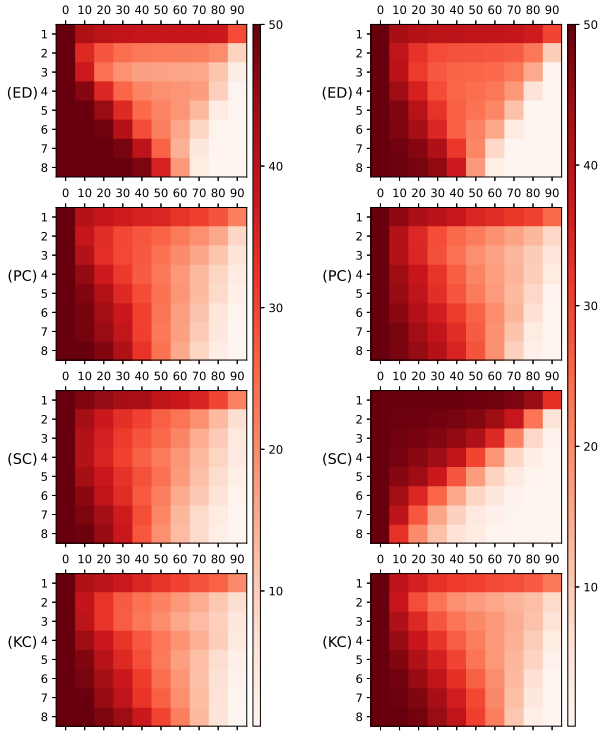
To verify the effectiveness of NS-IPMA method. The pruned DeepESNs were compared with a control experiment, the unpruned DeepESN, whose number of neurons in each reservoir is the same, the pruned DeepESN and the unpruned DeepESN were compared with equal layer number, equal total neruon number and same hyperparameters. The unpruned DeepESN is a standard benchmark that indicates the evolutionary characteristic of network performance by reducing network size. 90 percentage of neurons of random initalized DeepESNs were continuous pruned by different criterion based NS-IPMA methods (ED-IPMA, PC-IPMA, SC-IPMA, KC-IPMA) and non-criterion based IPMA,during pruning, networks were silhouetted and the performance were evaluated once 10 percentage of neurons had been pruned, these pruned groups were compared with unpruned DeepESN. All experiments were repeated for 20 times and all results were averaged through 20 independent replications.



(a) *MG* : 4×100 (b) *LZ* : 4×100

Figure 6. *Cont.*

 doi:10.20944/preprints202301.0533.v1



(c) $MG : 8 \times 50$ (d) $LZ : 8 \times 50$

Figure 6. The number of remaining neurons in each layer of peuned DeepESN, which was processed by different similarity estimation criteria based NS-IPMA methods. The vertical axis indicates layer index, the horizontal axis indicates percentage of pruned neurons in the initial total number of neurons, the mesh color indicates the number of remained neurons, the darker color indicates more neurons are reamained. (a):Initial 4 layer reservoirs with 100 neurons in each reservoir on *MG* dataset;(b):Initial 8 layer reservoirs with 50 neurons in each reservoir on *MG* dataset; (c):Initial 4 layer reservoirs with 100 neurons in each reservoir on *LZ* dataset;(d):Initial 8 layer reservoirs with 50 neurons in each reservoir on *LZ* dataset.

4.3. Results and disscusion

4.3.1. Hierarchical structure

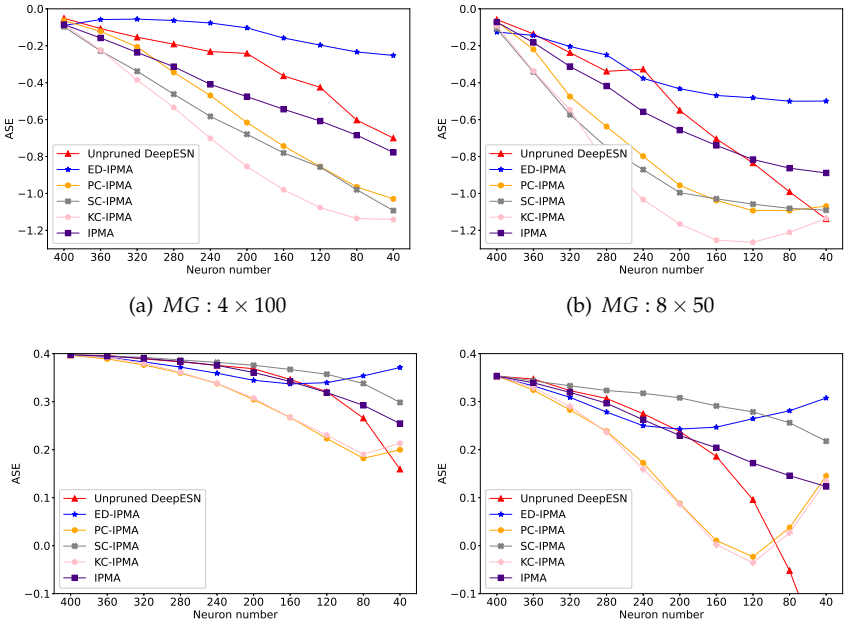
During pruning, the number of remaining neurons in each reservoir of pruned DeepESN, which was processed by different similarity estimation criterion based NS-IPMA, are shown in Figure 5 As NS-IPMA goes on, we observed significantly reduction of the neuron number in high layers at a later stage, the reason was because the discard policy, of which when a redundant pair of most similar neuron in different reservoirs were found, the neuron in higher layers would be pruned out. Lower layer reservoirs are the foundation of higher layer reservoirs, too few neurons in lower layers brings risk of insufficient information presentation and extracion in higher layers. Thus, the NS-IPMA methods have good network structure adaptability .

4.3.2. Reservoir diversity and error performance

The quantitative performance comparations (ASE, training NRMSE, and testing NRMSE) of unpruned DeepESN and DeepESN, which was pruned by IPMA and different similarity estimation criterion based NS-IPMA, are illustrated in Figures 7–9 The results on the two datasets were similar but different. On the whole, as reservoir size reduces, ASE will decrease and training NRMSE will increase, confirming that the model with more neurons has better information representation ability and better training effect. Initially, testing NRMSE is greater than training NRMSE, this phenomenon

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 January 2023 doi:10.20944/preprints202301.0533.v1

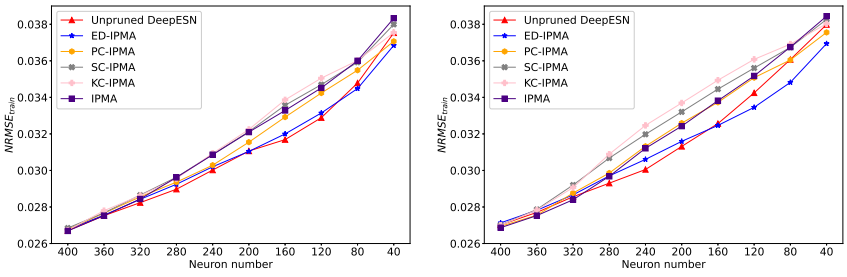
is called overfitting. As the reservoir size reduces, The training error went down and the overfitting is improved. confirming that the reducing the network size is a directly effective approach to improve generalization performance of the network. There is no doubt that pruning neurons will lead to reduction of network size, that is the reason why the unpruned DeepESN is tested as a benchmark. A successful pruning algorithm should outperform this benchmark. Furthermore, the minimum testing NRMSE condition is choosen to compare the extreme generalization performance of each experiment group, which was recorded in Table 1.



(a) $MG : 4 \times 100$ (b) $MG : 8 \times 50$

(c) $LZ : 4 \times 100$ (d) $LZ : 8 \times 50$

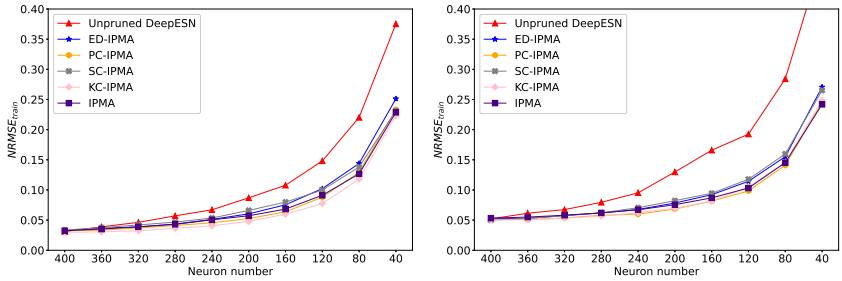
Figure 7. ASE Comparation of unpruned DeepESN and DeepESN pruned by IPMA, ED-IPMA, PC-IPMA, SC-IPMA, KC-IPMA. (a):Initial 4 layer reservoirs with 100 neurons in each reservoir on *MG* dataset;(b):Initial 8 layer reservoirs with 50 neurons in each reservoir on *MG* dataset; (c):Initial 4 layer reservoirs with 100 neurons in each reservoir on *LZ* dataset;(d):Initial 8 layer reservoirs with 50 neurons in each reservoir on *LZ* dataset.



(a) $MG : 4 \times 100$ (b) $MG : 8 \times 50$

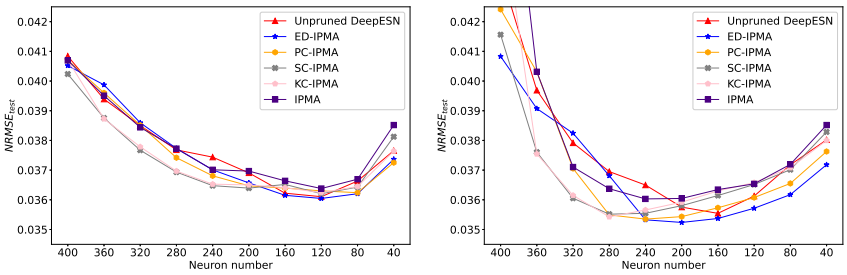
Figure 8. *Cont.*

 doi:10.20944/preprints202301.0533.v1

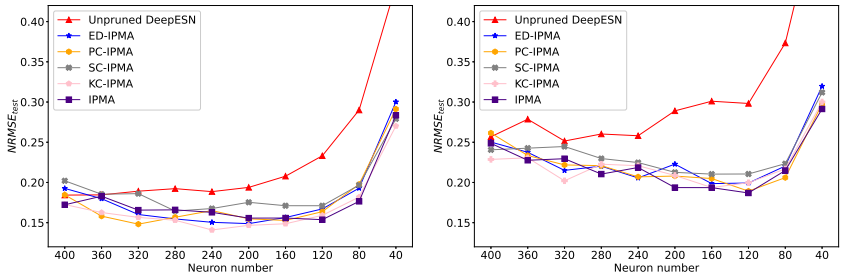


(c) $LZ : 4 \times 100$ (d) $LZ : 8 \times 50$

Figure 8. Training NRMSE Comparation of unpruned DeepESN and DeepESN pruned by IPMA, ED-IPMA, PC-IPMA, SC-IPMA, KC-IPMA. (a):Initial 4 layer reservoirs with 100 neurons in each reservoir on *MG* dataset;(b):Initial 8 layer reservoirs with 50 neurons in each reservoir on *MG* dataset; (c):Initial 4 layer reservoirs with 100 neurons in each reservoir on *LZ* dataset;(d):Initial 8 layer reservoirs with 50 neurons in each reservoir on *LZ* dataset.



(a) $MG : 4 \times 100$ (b) $MG : 8 \times 50$



(c) $LZ : 4 \times 100$ (d) $LZ : 8 \times 50$

Figure 9. Testing MRMSE Comparation of unpruned DeepESN and DeepESN pruned by IPMA, ED-IPMA, PC-IPMA, SC-IPMA, KC-IPMA. (a):Initial 4 layer reservoirs with 100 neurons in each reservoir on *MG* dataset;(b):Initial 8 layer reservoirs with 50 neurons in each reservoir on *MG* dataset; (c):Initial 4 layer reservoirs with 100 neurons in each reservoir on *LZ* dataset;(d):Initial 8 layer reservoirs with 50 neurons in each reservoir on *LZ* dataset.

 doi:10.20944/preprints202301.0533.v1

Table 1. Minimum testing NRMSE condition of each experiment group.

	Group	Unpruned DeepESN	Pruned DeepESN			
	Method	-	ED-IPMA	PC-IPMA	SC-IPMA	KC-IPMA	IPMA
$MG : (4 \times 100)$	$NRMSE_{test}$ (Mean)	0.036 105	**0.036 046**	0.036 240	0.036 206	0.036 243	0.036 380
	$NRMSE_{test}$ (Std.)	0.000 483	0.000 388	0.000 221	0.000 281	0.000 144	0.000 291
	Neuron number	120	120	80	120	120	120
$MG : (8 \times 50)$	$NRMSE_{test}$ (Mean)	0.035 544	**0.035 238**	0.035 350	0.035 516	0.035 427	0.036 030
	$NRMSE_{test}$ (Std.)	0.000 699	0.000 346	0.000 516	0.000 533	0.000737	0.000 338
	Neuron number	160	200	240	280	280	240
$LZ : (4 \times 100)$	$NRMSE_{test}$ (Mean)	0.184 285	0.148 812	0.152 775	0.164 455	**0.140 930**	0.155 667
	$NRMSE_{test}$ (Std.)	0.043 201	0.026 072	0.024 417	0.022 953	0.031 862	0.027 816
	Neuron number	400	200	160	280	240	200
$LZ : (8 \times 50)$	$NRMSE_{test}$ (Mean)	0.251 535	0.198 581	0.189 230	0.210 376	0.193 678	**0.186 933**
	$NRMSE_{test}$ (Std.)	0.044 484	0.027 631	0.024 995	0.033 404	0.029 164	0.024 092
	Neuron number	320	160	120	160	160	120

Bold values indicate the best validation performance of all methods.

 doi:10.20944/preprints202301.0533.v1

Similarly, from Figure 7, the DeepESN pruned by ED-IPMA maintains a minimum ASE loss as the number of neurons decreases, we suspect that there are some hidden relationships between Euler distance and Renyi's quadratic entropy; from Table 1, the pruned DeepESNs have a samller standard deviation of the minimum testing NRMSE compared to the unpruned DeepESN, indicating that NS-IPMA method has good robustness.

On MG dataset: from Figure 9a,b, In the early stage, the testing error of DeepESN pruned by SC-IPMA and KC-IPMA drops rapidly; later, DeepESN pruned by ED-IPNA and PC-IPMA out stand from crowd when the majority of neurons are pruned. From Table 1, although ED-IPMA achieved the best extreme generalization performance, the pruned DeepESNs have no obvious improvement on the mean value of minimum testing NRMSE compared to the unpruned DeepESN.

On LZ dataset: from Figures 8c,d and 9c,d, and Table 1, the taining error, testing error, and extreme generalization performance of pruned DeepESNs are all significantly improved compared with unpruned DeepESN, and the diversity of different similarity estimation criteria are not prominent, non-criterion based IPMA method perfomed as good as criterion based NS-IPMA methods.

In summary, all these experimental results showed that the NS-IPMA method is a successful approach to improving the generalization performance of DeepESN, which is specific in, under most circumstances of our designed experiments, the DeepESN pruned by NS-IPMA has better generalization performance than the standard unpruned DeepESN.

5. Conclusions

In our research, a new Iterative Pruning Merging algorithm was proposed to simplify the architecture of DeepESN. As to which neurons should be pruned out, four different similarity estimation criteria were attempted. The unpruned DeepESNs is a benchmark that indecates the evolutionary characteristic of network performance by reducing network size, the effectiveness of proposed methed was experimentally verified by comparing pruned DeepESNs with unpruned DeepESNs in the same network size. The results showed that these NS-IPMA methods have good network structure adaptability, and the DeepESNs pruned by NS-IPMA methed have better generalization performance and better robustness than unpruned DeepESNs, indicating that NS-IPMA methed is a feasible and superior approach to improving the generalization performance of DeepESN. The NS-IPMA method provides a novel approach for choosing the appropriate network size of DeepESN, one could start with a larger model than necessary reservoir size , then prune or merge some similar neurons to obtain a better DeepESN model, one could select a simple architecture with small computation requirements while keeping the testing error acceptable.

However, there are still some shortcomings in our work: First, the problem of how to choose the redundant neurons to be pruned out, or what the best neuronal similarity estimation criterion should be, remains unsolved. Second, only the hierarchical structure, the reservoir diversity, and the overall error performance are investigated, more evolutionary characteristics of different reservoirs, such as their spectral radius, resulting from the NS-IPMA method are not analyzed. Third, the effects of pruning and merging are not clearly distinguished. Further research on the NS-IPMA method is expected to be carried out.

 doi:10.20944/preprints202301.0533.v1

Appendix A. Hyperparameter tuning

Table A1. Hyperparameters applied under different initial resverior size on different datasets.

Dataset	*MG*	*MG*	*LZ*	*LZ*
Initial size	4×100	8×50	4×100	8×50
α	0.92	0.92	0.92	0.92
γ_r	0.8	0.8	0.8	0.8
${\gamma_i}^{*1}$	0.373 84	0.253 14	0.096 31	0.064 94
${\gamma_p}^{*2}$	0.211 36	0.241 62	0.335 51	0.347 11
λ	1×10^{-10}	1×10^{-10}	1×10^{-10}	1×10^{-10}

*1 Tuned by PSO in the range of $[1 \times 10^{-5}, 10]$; *2 Tuned by PSO in the range of $[0.1, 5]$.

As defined in Section 2.2, hyperparameters(α, γ_i, γ_r, γ_p and λ) play essential roles in the performance of DeepESN, so as in the successful application of NS-IPMA. α and γ_r affect stability of reservoirs, $\alpha = 0.92$ and $\gamma_r = 0.8$ are set to satisfy equation 6; λ affects the generalization performance, a small regularization facter $\lambda = 1 \times 10^{-10}$ is chosen to make the output weights better fit the training samples. γ_i adjusts the strength of input signal into the first layer of DeepESN, γ_p adjusts the strength of input signal into higher layers of DeepESN, thus, γ_i is optimized on the LI-ESN which is the first layer of DeepESN, after that γ_p is optimized on the DeepESN after higher layers hierarchically stacked on original LI-ESN. The tuned results are recorded in Table A1.

References

1. Gallicchio, C.; Micheli, A. Architectural richness in deep reservoir computing. *Neural Computing and Applications* **2022**. https://doi.org/10.1007/s00521-021-06760-7.
2. Jaeger, H.; Haas, H. Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication. *Science* **2004**, *304*, 78–80. https://doi.org/doi:10.1126/science.1091277.
3. Zhang, S.; He, K.; Cabrera, D.; Li, C.; Bai, Y.; Long, J. Transmission Condition Monitoring of 3D Printers Based on the Echo State Network. *Applied Sciences* **2019**, *9*. https://doi.org/10.3390/app9153058.
4. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Computation* **1997**, *9*, 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.
5. Che, Z.P.; Purushotham, S.; Cho, K.; Sontag, D.; Liu, Y. Recurrent Neural Networks for Multivariate Time Series with Missing Values. *Scientific Reports* **2018**, *8*. https://doi.org/10.1038/s41598-018-24271-9.
6. Xu, X.; Ren, W. Prediction of Air Pollution Concentration Based on mRMR and Echo State Network. *Applied Sciences* **2019**, *9*. https://doi.org/10.3390/app9091811.
7. Zhang, M.; Wang, B.; Zhou, Y.; Sun, H. WOA-Based Echo State Network for Chaotic Time Series Prediction. *Journal of the Korean Physical Society* **2020**, *76*, 384–391. https://doi.org/10.3938/jkps.76.384.
8. Zhang, M.; Wang, B.; Zhou, Y.; Gu, J.; Wu, Y. Prediction of Chaotic Time Series Based on SALR Model with Its Application on Heating Load Prediction. *Arabian Journal for Science and Engineering* **2021**, *46*, 8171–8187. https://doi.org/10.1007/s13369-021-05407-y.
9. Zhou, J.; Wang, H.; Xiao, F.; Yan, X.; Sun, L. Network traffic prediction method based on echo state network with adaptive reservoir. *Software-Practice & Experience* **2021**, *51*, 2238–2251. https://doi.org/10.1002/spe.2950.
10. Baek, J.; Choi, Y. Deep Neural Network for Predicting Ore Production by Truck-Haulage Systems in Open-Pit Mines. *Applied Sciences* **2020**, *10*. https://doi.org/10.3390/app10051657.
11. Gallicchio, C.; Micheli, A.; Pedrelli, L. Deep reservoir computing: A critical experimental analysis. *Neurocomputing* **2017**, *268*, 87–99. https://doi.org/10.1016/j.neucom.2016.12.089.
12. Gallicchio, C.; Micheli, A.; Silvestri, L. Local Lyapunov exponents of deep echo state networks. *Neurocomputing* **2018**, *298*, 34–45. https://doi.org/https://doi.org/10.1016/j.neucom.2017.11.073.
13. Gallicchio, C.; Micheli, A. Echo State Property of Deep Reservoir Computing Networks. *Cognitive Computation* **2017**, *9*, 337–350. https://doi.org/10.1007/s12559-017-9461-9.

14. Gallicchio, C.; Micheli, A. Deep Echo State Network (DeepESN): A Brief Survey. *ArXiv* **2017**, *abs/1712.04323*. https://doi.org/10.48550/arXiv.1712.04323.
15. Gallicchio, C.; Micheli, A. Why Layering in Recurrent Neural Networks? A DeepESN Survey. In Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. https://doi.org/10.1109/IJCNN.2018.8489368.
16. Thomas, P.; Suhner, M.C. A New Multilayer Perceptron Pruning Algorithm for Classification and Regression Applications. *Neural Processing Letters* **2015**, *42*, 437–458. https://doi.org/10.1007/s11063-014-9366-5.
17. Wang, H.; Yan, X. Improved simple deterministically constructed Cycle Reservoir Network with Sensitive Iterative Pruning Algorithm. *Neurocomputing* **2014**, *145*, 353–362. https://doi.org/10.1016/j.neucom.2014.05.024.
18. Jaeger, H.; Lukoševičius, M.; Popovici, D.; Siewert, U. Optimization and applications of echo state networks with leaky- integrator neurons. *Neural networks : the official journal of the International Neural Network Society* **2007**, *20 3*, 335–52. https://doi.org/10.1016/j.neunet.2007.04.016.
19. Yildiz, I.B.; Jaeger, H.; Kiebel, S.J. Re-visiting the echo state property. *Neural Networks* **2012**, *35*, 1–9. https://doi.org/https://doi.org/10.1016/j.neunet.2012.07.005.
20. Gallicchio, C.; Micheli, A. Architectural and Markovian factors of echo state networks. *Neural Networks* **2011**, *24*, 440–456. https://doi.org/https://doi.org/10.1016/j.neunet.2011.02.002.
21. Rodan, A.; Tino, P. Minimum complexity echo state network. newblock *IEEE Transactions on Neural Networks* **2011**, *22*, 131–144. https://doi.org/10.1109/TNN.2010.2089641.
22. Castellano, G.; Fanelli, A.; Pelillo, M. An iterative pruning algorithm for feedforward neural networks. *IEEE Transactions on Neural Networks* **1997**, *8*, 519–531. https://doi.org/10.1109/72.572092.
23. Islam, M.M.; Sattar, M.A.; Amin, M.F.; Yao, X.; Murase, K. A New Adaptive Merging and Growing Algorithm for Designing Artificial Neural Networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* **2009**, *39*, 705–722. https://doi.org/10.1109/TSMCB.2008.2008724.
24. Mackey, M.C.; Glass, L. Oscillation and chaos in physiological control systems. *Science* **1977**, *197 4300*, 287–9.
25. Maat, J.R.; Malali, A.; Protopapas, P. TimeSynth: A Multipurpose Library for Synthetic Time Series in Python, **2017**. https://github.com/TimeSynth/TimeSynth
26. Chao, K.H.; Chang, L.Y.; Xu, F.Q. Smart Fault-Tolerant Control System Based on Chaos Theory and Extension Theory for Locating Faults in a Three-Level T-Type Inverter. *Applied Sciences* **2019**, *9*. https://doi.org/10.3390/app9153071.