Article

Sparse Data-Driven Model for Monitoring of Industrial Process Based on Distributed Feature Extraction Neural Network

Zhen Li , Yibin Han , Yanxia Yang [*]

Posted Date: 4 July 2025

doi: 10.20944/preprints202507.0364.v1

Keywords: process monitoring model; sparse feature; sparse subspace clustering; collaborative optimization algorithm; distributed neural network

Article

Sparse Data-Driven Model for Monitoring of Industrial Process Based on Distributed Feature Extraction Neural Network

Zhen Li, Yibin Han and Yanxia Yang *

Address
* Correspondence: wuxl@bjut.edu.cn

Abstract

Modern industrial processes store a large amount of process data in the integration of subsystems or subprocesses, which creates conditions for data-driven models. However, due to a few features in various tasks possessing a direct correlation, there are many variables and complex relationships, which may result in incomplete input information. To address this problem, this paper proposes a task decomposition and feature integration-based distributed process monitoring model. Firstly, a sparse subspace clustering algorithm is introduced for task decomposition. This algorithm divides the original space into several interactive feature subspaces and allocates weights to quantify the contribution of the subtask simultaneously. Secondly, based on the divided features, a distributed framework for spatial feature integration is proposed. The framework constructs a differentiated parallel coding network by designing a structural self-organization mechanism, which achieves feature extraction and fusion of each subspace. Finally, a collaborative optimization algorithm is proposed to optimize the network parameters of each sub-model at the same time to ensure the accuracy of the model. To demonstrate the effectiveness of this data modeling method, we tested it in several benchmark data sets and a high-dimensional nonlinear system. The experimental results show that the model has better performance in data dimensionality reduction.

Keywords: *index terms*—process monitoring model; sparse feature; sparse subspace clustering; collaborative optimization algorithm; distributed neural network

---

I. INTRODUCTION

In the past decades, process monitoring model (PMM) has been widely used for variables monitoring, fault diagnosis, and performance evaluation in industrial processes [1–3]. With the development of acquisition equipment, automation, and transmission technology, a mass of irrelevant process variables are stored, industrial process monitoring model is confronted with the problem of feature sparsity [4–6]. The inflow of large-scale incomplete information will increase the model deviation and the difficulty of parameter training [7–9]. Therefore, it is urgent to develop a multitask clustering (MTC) algorithm that 1) not only automatically mines the potential relationship of variables from the data 2) but also reduces the size of the problem and improves the efficiency of model processing. To achieve the above goal on MTC, we need to resort to two research areas—multitask clustering and model design.

To divide the feature space, many clustering algorithms have been proposed. Meanwhile, partitioning methods such as k-means are probably one of the most frequently used techniques. As widely reported in the literature, however, the performance of this kind of algorithm is unstable due to the influence of the initially selected central sample [10–12]. To improve the algorithm performance, there have been many research proposals [13,14]. In [13], Modha et al. proposed a feature weighted k-means clustering algorithm, to be the one that yields the clustering that simultaneously minimizes the average within-cluster dispersion and maximizes the average between-cluster dispersion along all the feature spaces. In [14], Luo et al. proposed a spatial constrained k-means clustering algorithm, which adopted the spatial constraints into the hierarchical K-means clusters on each image level. This kind of algorithm can only get a single spherical partition of the data input space. To further mine data information, some more complex clustering methods are proposed [15,16]. For example, Louhichi et al. proposed a density-based algorithm for discovering clusters to obtain the local density change in a large database with noise [15]. Cheng *et al*. proposed a multistage random sampling clustering algorithm based on fuzzy c-means, which effectively improves computational efficiency by significantly reducing the clustering time [16]. However, all of the proposed algorithms in [13–16] are aimed at linear separable data space, and it is hard to find suitable clustering contours and stable clustering results for indivisible data. By introducing a kernel function, the algorithm based on the kernel function can project the linearly inseparable data in the input space into the high-dimensional space to make it separable [17,18]. In [17], Chiang et al. extended the support vector clustering to an adaptive cell growing model that maps data points to a high-dimensional feature space through a desired kernel function. In [18], Fan et al. designed a self-adaptive kernel k-means algorithm, which can adjust the kernel parameter automatically according to the data structure. The above clustering methods all contain division criteria based on distance measurement. With the increase of data dimensions, the data are almost equidistant, and the distance metric becomes meaningless. Manifold Learning considers that high-dimensional data will exhibit dense aggregation in a low-dimensional space. Therefore, the subspace clustering method (SCM) is derived to find the low-dimensional subspace in high-dimensional data [19,20]. For example, Kang et al. proposed a unified multi-view subspace clustering model that incorporates the graph learning from each view, the generation of basic partitions, and the fusion of consensus partitions [19]. Xu et al. adopt soft subspace clustering to solve the problem of rule redundancy of fuzzy systems in high-dimensional data [20]. Several articles have mentioned that SCM provides a variety of perspectives for feature extraction of sparse data to ensure the maximum use of input information [20–23].

After the feature space is divided, the original problem becomes a multi-tasking problem. There are many models based on local-global ideas that can be used to solve the above problem [24–26]. For example, Koker et al. proposed a parallel feed-forward neural network structure is used in the prediction of Parkinson's Disease [24]. Gupta et al. proposed a new technique to train deep neural networks over several data sources, which allowed for deep neural networks to be trained using data from multiple entities in a distributed fashion [25]. Cao et al. proposed a fuzzy rough neural network via distributed parallelism, where each model is transformed into a multi-objective optimization problem [26]. The subnets of these models are often isomorphic and are only suitable for cases where the feature subset is similar [27–30]. MNN adopts the divide-and-conquer strategy, divides the main tasks into several simple subtasks, and obtains heterogeneous subnet modules according to different tasks, thus improving the overall generalization performance [31,32]. In [31], Li et al.（引言里的et al.是不是都要斜体啊？） introduced an enhanced feature-weighted modular neural network (MNN), in which each RBF subnetwork was independently constructed, and the final output is obtained by aggregating the outputs of all subnetworks through a weighted summation, where the weights reflected the contribution of each subnetwork. In [32], Valdez et al. proposed a new hybrid approach combining particle swarm optimization and genetic algorithms, which used fuzzy logic to integrate the results of subnetwork modules in MNN for face recognition. In the follow-up research, a series of methods have been proposed that MNN can independently generate heterogeneous subnetworks to solve online problems [33,34]. For example, Qiao et al. proposed an online self-adaptive MNN for time-varying systems, which adopted a single-pass subtractive cluster algorithm to divide the input space and used a fuzzy strategy to integrate the results of subnetworks [33]. Loo et al. proposed a novel self-regulating algorithm to generate an optimum growing multi-experts network structure, which adopted a modified fully self-organized simplified adaptive resonance theory and self-adaptive learning rates for gradient descent learning rules to dynamically grow and prune the MNN's structure at a fixed step [34]. However, MNN has the problem of long learning time due to the slow convergence rate of gradient decline [35–38]. Moreover, the preset output weight makes it difficult to obtain an accurate prediction output according to the change of input distribution [39].

In this paper, to fully exploit sparse data information and obtain stable modeling performance, a task decomposition and feature integration-based distributed process monitoring model is proposed to realize the feature fusion of multi-task subspace. Our proposed PMM has the following properties: Firstly, an improved sparse subspace clustering algorithm (ISSC) constructs the affinity matrix by the distance between the sample and the subspace to obtain dense low-rank subspaces. This ISSC algorithm can retain the correlation information of features while efficiently feature division is achieved. Secondly, a distributed framework for spatial feature integration constructs differentiated coding networks in parallel, and designs model evaluation indexes according to subspace sparsity of each coding network, which can achieve feature extraction and integration of each subspace. Thirdly, a collaborative optimization algorithm optimizes the network parameters of each sub-model at the same time to ensure the accuracy of the model, which can maintain the accuracy of PMM.

The rest of this paper is organized as follows. Section II briefly reviews and discusses the basics of SSC. Section III details the proposed PMM framework, including ISSC and the distributed neural network. Then, the parameter update process of the distributed neural network is given in Section IV. Section V reports some experimental results of the proposed PMM, which demonstrate some merits in learning speed and modeling accuracy against other existing methods. Section VI concludes this paper with some remarks.

II. Problem Formulation and Preliminaries

Industrial processes have accumulated a large number of process variables, most of which have no significant correlation. Sparse coding advocates building a more concise feature representation [40,41]. The data set is divided into meaningful blocks to highlight the local information, so that these blocks are not related to each other to the maximum extent, so as to obtain dense feature representation.

Given a feature **x** and a basis pool **U** $=[u_1, u_2, ..., u_k]$, sparse coding aims at sparsely and linearly reconstructing the feature to be encoded, $\mathbf{x}=v_1u_1+v_2u_2+\cdots+v_ku_k$. Here, sparseness means only a small fraction of elements in v are non-zero. The optimization problem of sparse coding can be calculated as follows

$$\min_{v} \|v\|_0 \quad st. \quad \mathbf{x} = \mathbf{U}\mathbf{v}, \tag{1}$$

where, $\|v\|_0$ means the number of non-zero elements in v. However, the minimization of ℓ_0 norm is an NP hard problem. In many practical applications, reconstruction error is inevitable and often difficult to estimate in advance. Therefore, it is beneficial to jointly optimize both the sparsity of the coefficients and the reconstruction error. Accordingly, the objective function of sparse coding can be reformulated as follows

$$\min_{U,v} \|x - Uv\|^2 + \lambda \|v\|_1 \quad st. \quad u_m \le 1, \tag{2}$$

where, the first term of Eq. (2) is the reconstruction error, and the second term is used to control the sparsity of the sparse codes v. And λ is the tradeoff parameter used to balance the sparsity and the reconstruction error.

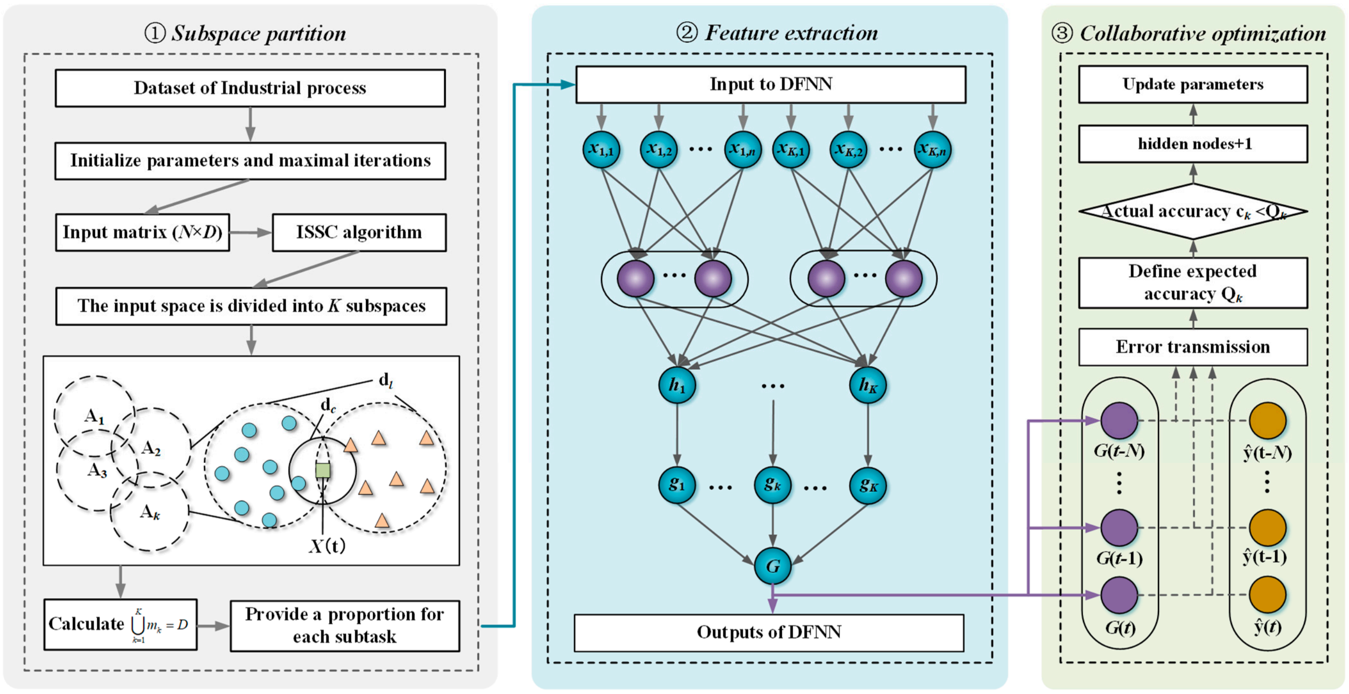

Fig. 1. Structure of ISSC-DNN

III. Architecture of ISSC-DFNN

Under a global-local framework, ISSC-DFNN is composed of four primary components: the input layer, Subspace partition, feature extraction, and feature integration, as illustrated in Fig. 1. The structure and functionality of each layer are detailed as follows

Input: This component first loads the dataset and then employs Eq (3) to evaluate the relative contribution of each process variable, thereby removing redundant features. An input matrix X=[$\mathbf{x}_1$,$\mathbf{x}_2$, ..., $\mathbf{x}_D$] of size N×D, where N denotes the number of samples and D denotes the number of features. The dth feature vector is denoted as $\mathbf{x}_d$=[$x_{d,1}$, $x_{d,2}$, ..., $x_{d,N}$]$^{\mathrm{T}}$(d =1, 2, …., D).

Subspace partition: In this block, by using the idea of manifold learning, the sparse original space is divided into several interactive low-dimensional subspaces. An enhanced sparse subspace clustering algorithm is proposed to partition the input matrix X （这个地方 X 好像是个矩阵，是不是需要加粗表示啊？）into K feature subsets, denoted as $\mathbf{A}_k$ (k = 1, 2, …, K), where each $\mathbf{A}_k$ is an N×m_k matrix and m_k indicates the number of features in the kth subset. These feature subsets are subsequently assigned to their corresponding subnetworks for further processing.

Feature extraction: The subnetwork layer in ISSC-DFNN consists of K feature extraction subnetworks, where K corresponds to the number of identified feature subspaces. Each subnetwork is adaptively constructed using the ErrCor algorithm and is responsible for processing a specific group of features in parallel.

Feature integration: The primary role of this layer is to fuse the outputs from multiple subnetworks via a weighted summation mechanism, with the integration weights determined based on the Sparse Similarity Index.

IV. Algorithm Design of ISSC-DNN

In this section, we first introduce a sparse subspace task decomposition clustering algorithm, which segments the original space into several mutually associated feature subspaces, with corresponding weights assigned to characterize the importance of each subtask, and then uses a structured self-organizing mechanism, differentiated parallel coding network DFNN to realize feature extraction and fusion of each subspace. Subsequently, a global gradient descent algorithm is employed to simultaneously optimize the parameters of all subnetworks, thereby ensuring the overall model accuracy.

A. The subspace partition method of spectral clustering

The subspace clustering algorithm selects several dimensions into space groups in the initial dimensions, which is not simply cut, and can avoid information loss in the process of dimension reduction. However, the contribution of different dimensions to clustering is not the same, and the dimensions may even be related to each other. This means that the dimension of the subspace cannot be obtained adaptively when each dimension of the subspace is assigned the same weight. Given a data matrix $\mathbf{X}\in\mathbb{R}^{N\times D}$, the soft subspace clustering (SSC) algorithm performs decomposition along either the row or column dimension, assigning a membership value to each row or column. Based on these membership values, the algorithm determines the cluster affiliation of each row or column. This allows some instances or features to be associated with multiple clusters simultaneously, resulting in overlapping, soft partitions that reflect compatibility across clusters. Focusing on row-wise decomposition, the corresponding objective function of the SSC algorithm is defined as follows

$$\begin{gathered} J(T)=\sum_{k=1}^{K}\sum_{j=1}^{N}u_{jk}{}^{m}\sum_{i=1}^{D}w_{ik}(x_{ij}-v_k{}')^2, \\ 0\le u_{jk}\le 1,\sum_{k=1}^{K}u_{jk}=1, 0\le w_{ik}\le 1,\sum_{k=1}^{K}w_{ik}=1, \end{gathered} \tag{3}$$

where K denotes the number of clusters, while N and D represent the number of rows and columns of the data matrix X, respectively. The matrix $\mathbf{v}=[v_1, v_2, ..., v_k, ..., v_K]$ contains the centers of all clusters, and $\mathbf{W}=[\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_D]$ is the weight matrix, where each column vector reflects the relative importance of features within a given cluster. $\mathbf{U}=[u_1, u_2, ..., u_N]$ denotes the membership matrix for the data samples, and mmm is the fuzzy weighting exponent. The three components of the objective function quantify the intra-cluster compactness, the entropy of feature weights, and the inter-cluster separation, respectively. The primary objective of the algorithm is to minimize intra-cluster compactness while maximizing inter-cluster separation by iteratively updating **v**, **W**, and **U** under a fixed number of clusters K.

Given an $N\times D$ data matrix X, K sample points are randomly given as the clustering center point $\mathbf{v}=[v_1, v_2, ..., v_k, ..., v_K]$. The membership matrix of the sample is obtained as $\mathbf{U}=[u_1, u_2, ..., u_j, ..., u_N]$, $u_j=[u_{j1}, u_{j2}, ..., u_{jk}, ..., u_{jK}]$, u_{jk} represents the membership degree of each sample x_{ij} belonging to the kth category

$$u_{jk}=\frac{d_{jk}{}^{\frac{1}{1-m}}}{\sum_{k=1}^{K}d_{jk}{}^{\frac{1}{1-m}}}, \tag{4}$$

where, i=1, 2, ..., D, j=1, 2, ..., N, k=1, 2, ..., K, D and N are the number of relevant variables and the number of samples respectively, K represents the total number of categories, m>1 is the fuzzy parameter, determined by the input N and D

$$m=\left\lceil\frac{\min(N,D-1)}{\min(N,D-1)-2}\right\rceil, \tag{5}$$

d_{jk} is the Euclidean distance of each sample x_{ij} relative to the k-th cluster center

$$d_{jk}=\sum_{i=1}^{D}(x_{ij}-v_k)^2 \tag{6}$$

Update the existing cluster center $\mathbf{v}' = [v_1', v_2', ..., v_K']$, and calculate the cluster center v_k' of the k-th category as

$$v_k{}'=\frac{\sum_{j=1}^{N}u_{jk}{}^{m}x_{ij}}{\sum_{j=1}^{N}u_{jk}{}^{m}} \tag{7}$$

According to the updated clustering center, the sample membership degree is recalculated by Eq. (1) and Eq. (3), and the contribution degree of each line feature in the original dataset belonging to the kth category is denoted as $\mathbf{W}=[\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_i, ..., \mathbf{w}_D]^T$, $\mathbf{w}_i= [w_{i1}, w_{i2}, ..., w_{ik}, ..., w_{iK}]$, where w_{ik} indicates the contribution of the $\mathbf{x}_i$ of line i to category k

$$w_{ik} = \sum_{j=1}^{N} u_{ik}{}^{m} (x_{ij} - v_k{}')^2 \tag{8}$$

To determine the number of cluster category K, we add a cluster center in each iteration, then recalculate sample membership until the objective function is minimal.

In this study, the optimal number of clusters is determined by selecting the value that minimizes the objective function within the interval [2, 2log(D)], where D represents the number of input features. This strategy ensures that, even in high-dimensional input spaces, the number of resulting subtasks remains within a manageable range. A detailed description of the overall algorithm is provided in Table 1.

Table 1. The ISSC based feature partitioning algorithm.

```
1: C=1
2: Initialize the parameters and set the maximum number of iterations to Max.
3: Repeat:
4: C=C+1;
5: Arbitrarily initialize cluster centers matrix V(0), initialize the weight matrix W(0).
6: Repeat:   iter = iter + 1;
7: Calculate the membership matrix U(iter) according to Eq. (4);
8: Calculate the weight matrix W(iter) according to Eq. (6);
9: Until ||V(iter)-V(iter-1)||<ζ or iter = Max.
10: Calculate the JIESSC by Eq. (2);
11: Until C = 2log(D); D is the number of input features.
12: Calculate the minimum value of J and the optimal clusters K.
```

B. Construction of subnetwork

1) Structure of subnetworks

In the following, we describe the DFNN layer by layer as a whole. The front network is composed of a distributed coding network, which is used to integrate redundant features in each subspace, and a corresponding self-organization strategy is designed to ensure that it can obtain different coding structures for each subspace. The latter network is responsible for feature expression for a regular FNN. The training of the whole network is also carried out separately into the two parts.

For the K subspaces $\mathbf{A}_k$ with the size of $N\times P_k$, he number of input neurons for the corresponding network is P_k (k=1, 2, …, K). Assuming there are h_k neurons in the hidden layer of the kth subnetwork, an auto-encoder maps it to a hidden representation h through an encoder function f as

$$h_k = f(\mathbf{W}_k + b_k) \tag{9}$$

with two parameters, i.e., weight matrix $\mathbf{W}_k \in \mathbb{R}^{n\times p}$ and the bias vector $b_k \in \mathbb{R}^n$. The hidden representation h can be mapped back to x, which is a reconstructed vector by the decoder function, as

$$\hat{x} = g(\mathbf{W}_k' + b_k') \tag{10}$$

where f is the sigmoid function, and g is the identity function. The cost function is mean squared error, because it is an appropriate choice for real-valued pixel intensity inputs.

2) Structure of subnetworks

Let Q_k denote the number of neurons in the hidden layer of the kth subnetwork. The output of the q_k th neuron (q_k =1, 2, ..., Q_k) is computed as follows

$$\boldsymbol{\phi}_j = \prod_{i=1}^{P_k} e^{-((x_i - c_{ij})^2 / 2\boldsymbol{\sigma}_{ij}^2)} = e^{-\sum_{i=1}^{n}((x_i - c_{ij})^2 / 2\boldsymbol{\sigma}_{ij}^2)} \qquad (11)$$
$$i = 1, 2, ..., P_k,\ j = 1, 2, ..., Q_k$$

where $\mathbf{x}$=[x_1, x_2,…, x_{P_k}] is the input of RBF layer, $\boldsymbol{\sigma}_j$=[σ_j^1, σ_j^2,…,σ_j^{Pk}] and $\mathbf{c}_j$=[c_j^1, c_j^2,…, c_j^{Pk}] are the vectors of widths and centers of the jth RBF neuron, respectively, v_j is the output value of the jth neuron, and Q_k is the number of neurons in this layer.

The output is clarified using the gravity method

$$y_k = \mathbf{W}^{'}\phi \qquad (12)$$

where

$$\mathbf{W}^{'} = \left[w_1, w_2, \cdots, w_{Q_k} \right] \qquad (13)$$

$\mathbf{W}^{'}$ is the parameter matrix, $\mathbf{W}^{'}$≜[w_1, w_2, …, w_{Qk}] are the weights between the qth neuron in the hidden layer and the output layer, y_k is the sole neuron in the output layer and can be calculated as

$$g_k = \mathbf{W}'\boldsymbol{\phi} = \sum_{j=1}^{Q_k} w_j e^{-\sum_{i=1}^{P_k}((x_i - c_{ij})^2 / 2\boldsymbol{\sigma}_{ij}^2)} \qquad (14)$$

Each subnet generates an attribute fusion feature. The output of all subnets should be normalized before feature integration

$$G = \frac{g_k}{\sum_{k=1}^{K} g_k} = \frac{w_j e^{-\sum_{i=1}^{P}((x_i - c_{ij})^2 / 2\boldsymbol{\sigma}_{ij}^2)}}{\sum_{k=1}^{K} w_j e^{-\sum_{i=1}^{P}((x_i - c_{ij})^2 / 2\boldsymbol{\sigma}_{ij}^2)}} \qquad (15)$$
$$k = 1, 2, \cdots, K,$$

where G is the output of the network.

3) Self-constructing of subnetworks

Through the subspace construction process, we obtain K independent subspace feature subsets $\mathbf{A}_k$. However, there is a highly collinear relationship between the features within these subspaces, and there is some information redundancy in the model input. It is necessary to construct an incomplete neural network structure (h_k<P_k) for more concise feature representation. In this paper, a self-organizing strategy based on pruning is proposed to ensure that DFNN codes differently according to different subspace inputs. Before introducing the self-organizing mechanism, the error of the kth subnetwork is defined as

$$e_k = \frac{1}{2} \times \left\| \hat{x}_i \text{ - } g(h(x_i)) \right\|, \qquad (14)$$

where x_i and x_i are the qth desired output and actual output of the output layer, respectively.

It is assumed that each hidden layer neuron is activated with a certain probability, and the hidden layer neurons are independent of each other. In the kth subspace, the average activation value of the jth neuron in the hidden layer of the subnet is expressed as

$$\hat{\boldsymbol{\rho}}_j = \frac{1}{N} \sum_{i=1}^{N} h_j(x^{(i)}) \qquad (15)$$

where N represents the number of samples in the kth subspace. The sparsity of neurons in the hidden layer can be calculated by

$$\boldsymbol{\mu}_j = \frac{\hat{\boldsymbol{\rho}}_j}{\hat{\boldsymbol{\rho}}_{\max}} \qquad (16)$$

where ρ_{max} represents the maximum average activation value and μ_0 represents the sparse parameter. If the target accuracy is not reached within the predefined maximum number of epochs, or if sparse nodes are detected, the corresponding hidden neurons are pruned, as illustrated in Eq (15)

$$c_{Q_k+1} = \left\{A_{k,n_{\max}}, \forall A_{k,n_{\max}} = \text{argmax}(|e_k|)\right\} \tag{17}$$

where e_k denotes the error vector of the kth subnetwork, with the width parameter initialized as σ_{Qk+1}=1, and the output weight set to l_{Qk+1}=1. In this approach, one new hidden neuron is added to the network during each training epoch. The algorithm terminates once the subnetwork achieves the target accuracy or when no sparse nodes are detected.

C. Collaborative optimization algorithm

In order to optimize the parameters of the subnetworks simultaneously, the comprehensive loss function is used for a unified solution. All subnetworks were centrally prepared at a fusion center, and then the global variable $\boldsymbol{\beta}^*$ could be obtained by solving the following problem.

$$u_i(\beta_i) = \frac{1}{2}\|\beta_i\|^2 + \frac{1}{2}\sum_{k=1}^{K} a_k \|\delta_k(t)\|^2 \tag{16}$$

where a_k is a scale factor to keep the loss functions of the front and back networks at the same range, δ is the partial derivatives of the loss to y, and g are the corresponding errors of their respective outputs.

$$\delta_k(t) = \frac{\partial L}{\partial \mathrm{y}_k(\mathrm{t})} = a_k\left(y_d(t) - y_k(t)\right) \tag{17}$$

Accordingly, the global objective can be restated as

$$u(\beta) = \sum_{i=1}^{N} u_i(\beta_i) \tag{18}$$

enabling all the nodes to cooperatively solve the problem. As has been shown previously, we next derive the updating formulation of the estimate $\beta_i(k)$and the initial state $\beta_i(0)$.

According to the expression in (16), we can verify that the difference between the gradients ∇u_i, evaluated at $\beta_i(k+1)$

and $\beta_i(k)$, respectively, can be given by

$$\begin{cases} \beta_i(k+1) = \beta_i(k) + \dfrac{\gamma}{VC}\displaystyle\sum_{j=1}^{N} a_{ij}(\beta_j(k) - \beta_i(k)) \\ \beta_i(0) = a_i\delta_i(t) \end{cases} \tag{19}$$

The positive parameter γ is chosen appropriately in (0, γ_{max}), where

$$\gamma_{max} = \frac{2VC}{\max\{\lambda_{max}(\delta_i)\}_{i=1}^{V}} \tag{20}$$

where

$$\beta_i(t) = \begin{bmatrix} A_i(t) \\ B_i(t) \\ C_i(t) \end{bmatrix} \tag{21}$$

$$A_i(t) = \begin{bmatrix} \mathbf{w}_1^1(t) \cdots \mathbf{w}_i^1(t) \cdots \mathbf{w}_P^1(t) \\ \vdots \quad \ddots \quad \vdots \\ \mathbf{w}_1^K(t) \cdots \mathbf{w}_i^K(t) \cdots \mathbf{w}_P^K(t) \end{bmatrix} \tag{22}$$

Table 2. Collaborative optimization algorithm.

```
1: T=1, Set the maximum number of iterations to T0;
2: Initialize expected accuracy $Q_k$, global variables $\beta*$, and neural network parameters $c_j^{Pk}$, $\sigma_j^{Pk}$ and $w_{Qk}$;
3: Repeat:
4: Repeat:   T=T+1;
5: Calculate the output of each neural subnet according to Eq (14);
6: Calculate the output error $\delta_k$ of each subnet according to Eq (17);
7: The global variable $\beta_i$ of each subnet can be obtained according to Eq (16);
8: Update subnet parameters by according to (18);
9: Until T> T0
10: The accuracy $c_{Qk}$ of the network is calculated according to Eq (15).
11: hidden nodes = hidden nodes + 1;
7: Until $c_{Qk}$ < $Q_k$;
```

$$B_i(t)=\begin{bmatrix} \boldsymbol{c}_1^1(t) & \cdots & \boldsymbol{c}_j^1(t) & \cdots & \boldsymbol{c}_P^1(t) \\ \vdots & & \ddots & & \vdots \\ \boldsymbol{c}_1^K(t) & \cdots & \boldsymbol{c}_j^K(t) & \cdots & \boldsymbol{c}_P^K(t) \end{bmatrix} \tag{23}$$

$$C_i(t)=\begin{bmatrix} \sigma_1^1(t) & \cdots & \sigma_j^1(t) & \cdots & \sigma_P^1(t) \\ \vdots & & \ddots & & \vdots \\ \sigma_1^K(t) & \cdots & \sigma_j^K(t) & \cdots & \sigma_P^K(t) \end{bmatrix} \tag{24}$$

where $\boldsymbol{\Delta}(t)$ is a unified parameter matrix, which is composed of front network parameter matrix $A(t)$ and post network parameter matrix $B(t)$, η is the adaptive learning rate, and $0<\mu<1$ is the learning rate adjustment parameter. $G(t)$ is the gradient vector.

According to the chain rule, the elements of the Jacobian matrix are calculated to be

$$\frac{\partial L}{\partial w_{ik}^{(1)}(t)}=\frac{\partial L}{\partial z_k}\frac{\partial z_k}{\partial h_k}\frac{\partial h_k}{\partial w_{ik}^{(1)}(t)}=z_k\left(1-z_k\right)\times w_k^{(2)}\delta_A\times x_i \tag{25}$$

$$\frac{\partial L}{\partial \boldsymbol{c}_j(t)}=\begin{bmatrix} \frac{\partial L}{\partial c_{1j}(t)} & \frac{\partial L}{\partial c_{2j}(t)} & \cdots & \frac{\partial L}{\partial c_{mj}(t)} \end{bmatrix} \tag{26}$$

$$\frac{\partial L}{\partial c_{ij}(t)}=-\frac{2w_j(t)\times v_i(t)\times\left[x_i(t)-c_{ij}(t)\right]}{\sigma_{ij}(t)} \tag{27}$$

$$\frac{\partial L}{\partial \boldsymbol{\sigma}_j(t)}=\begin{bmatrix} \frac{\partial L}{\partial \sigma_{1j}(t)} & \frac{\partial L}{\partial \sigma_{2j}(t)} & \cdots & \frac{\partial L}{\partial \sigma_{mj}(t)} \end{bmatrix} \tag{28}$$

$$\frac{\partial L}{\partial \sigma_{ij}(t)}=-\frac{w_j(t)\delta_B\times v_i(t)\times\left\|x_i(t)-c_{ij}(t)\right\|^2}{\sigma_{ij}^2(t)} \tag{29}$$

The parameter update amount of each subspace is calculated from Eqs. (19)-(29).

V. Simulation Studies

In this section, the performance of ISSC-DFNN was tested through several benchmark tasks and a real-world application dataset in an engineering application. Besides, ISSC-DFNN is compared with some other existing methods. All the simulations were programmed with MATLAB version 2016 in the same PC environment.

A. *Experimental datasets*

1) **Benchmark problems**

To assess the modeling performance of ISSC-DFNN, four benchmark datasets from the UCI Machine Learning Repository were selected for evaluation, namely Boston Housing, Auto Prices, Residential Building, and Abalone. Regression models were constructed using the input and output features of each dataset. Detailed information about these datasets is provided in Table 3. For comprehensive descriptions of each feature, please refer to the UCI Machine Learning Repository.

Table 3. Information for FOUR UCI datasets.

Dataset	Training samples	Testing samples	Input dimensions	Output dimension
Boston Housing	350	150	13	1
Auto Price	120	50	14	1
Abalone	1500	500	8	1
Residential building	350	150	107	1

2) **ETP in wastewater treatment**

Effluent total phosphorus (ETP) serves as a key indicator in sewage treatment, reflecting whether the discharged wastewater complies with regulatory standards. Accurate prediction of ETP is vital for the real-time control of the sewage treatment process. In this study, practical data were gathered from a sewage treatment plant in Beijing, and the ISSC-DFNN model was employed to perform the predictions. As shown in Table 4, the dataset of plant variables provides 23 input variables $X = \{X_1,\ldots, X_{23}\}$, two different monitoring target variables Y_d, and 1200 samples. These variables are composed of some process variables that are beneficial to monitoring, e.g., T, DO, ORP, MLSS, NO_3-N. There are also some irrelevant or weak correlation variables. In order to unify the time series of online data and laboratory data, the sampling frequency of all parameters was 10 minutes. After removing abnormal data, 1,200 samples from September 1, 2019, to October 31, 2019, will be obtained for normalization processing. There are 700 data samples in each data set, which consists of 600 training data and 100 test data.

Table 4. Input variables of ETP dataset.

X_1	Inlet flow	X_{13}	BOD
X_2	Temperature	X_{14}	Influent oil
X_3	ORP1	X_{15}	Effluent oil
X_4	ORP2	X_{16}	Influent ammonia
X_5	MLSS1	X_{17}	Influent colourity
X_6	NO3-N	X_{18}	Effluent colourity
X_7	NH4-N	X_{19}	Influent PH
X_8	DO_1	X_{20}	Effluent PH
X_9	ORP_3	X_{21}	Suspended Solid
X_{10}	$MLSS_2$	X_{22}	Influent nitrogen
X_{11}	NO3-N	X_{23}	Influent phosphate
X_{12}	DO_2	Y_d	ETP

B. *Experimental setup*

To enhance the operational efficiency of the ISSC algorithm within the DFNN framework, the parameters γ and η are chosen as 10 and 0.01, respectively. The parameter t is typically set to a relatively small value within the range [1,2] to ensure stable performance of the IESSC algorithm, in this study, it is empirically set to 1.5. The approximation performance of the model is evaluated using the root mean square error (RMSE) and the average percentage error (APE) between predicted and desired output, which is shown as (30) and (31):

$$RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(y_d(t) - g(t))^2}, \tag{30}$$

$$APE = \frac{1}{N}\sum_{t=1}^{N}\frac{\|y_d(t) - g(t)\|}{\|y_d(t)\|} \times 100\%, \tag{31}$$

where N denotes the sample size, and y_d and g represent the desired and actual outputs for the nth sample, respectively. To reduce the effect of randomness, each experiment corresponding to a dataset is independently repeated 10 times, and the average RMSE and APE values are reported as the final results.

To evaluate the superiority of ISSC-DFNN, we focus on the influence of distributed neural network on the process monitoring model. The modeling results are compared with other mainstream feature extraction algorithms: the models considered include a modular neural network with adaptive feature partitioning (FC-AMNN), the traditional modular neural network (TMNN), and an online self-organizing modular neural network (OSAMNN). Detailed descriptions of each model are provided in Table 5.

Table 5. Comparison results with other models for UCI benchmark problems.

Algorithm	Dataset A: Boston Housing				Dataset B: Auto Price			
	Training RMSE	Testing RMSE	Testing APE	Subnetworks	Training RMSE	Testing RMSE	Testing APE	Subnetworks
Prop.	**0.0759**	**0.0741**	**0.1375**	**2**	**0.0738**	**0.0742**	**0.1729**	**2**
FC-AMNN	0.0767	0.0748	0.1396	4	0.0752	0.0766	0.1781	2
TMNN	0.0875	0.0906	0.1684	2	0.0872	0.0883	0.1968	2
OSAMNN	0.0927	0.1006	0.1734	8	0.0785	0.0810	0.1892	5

Algorithm	Dataset C: Abalone				Dataset D: Residential building			
	Training RMSE	Testing RMSE	Testing APE	Subnetworks	Training RMSE	Testing RMSE	Testing APE	Subnetworks
Prop.	**0.0739**	**0.0741**	**0.1721**	**2**	**0.0987**	**0.0962**	**0.467**	**3**
FC-AMNN	0.0774	0.0786	0.1797	2	0.0994	0.1062	0.4963	4
TMNN	0.0823	0.0886	0.1895	2	0.1152	0.1204	0.5397	3
OSAMNN	0.0815	0.0879	0.1838	7	0.1074	0.1126	0.5124	4

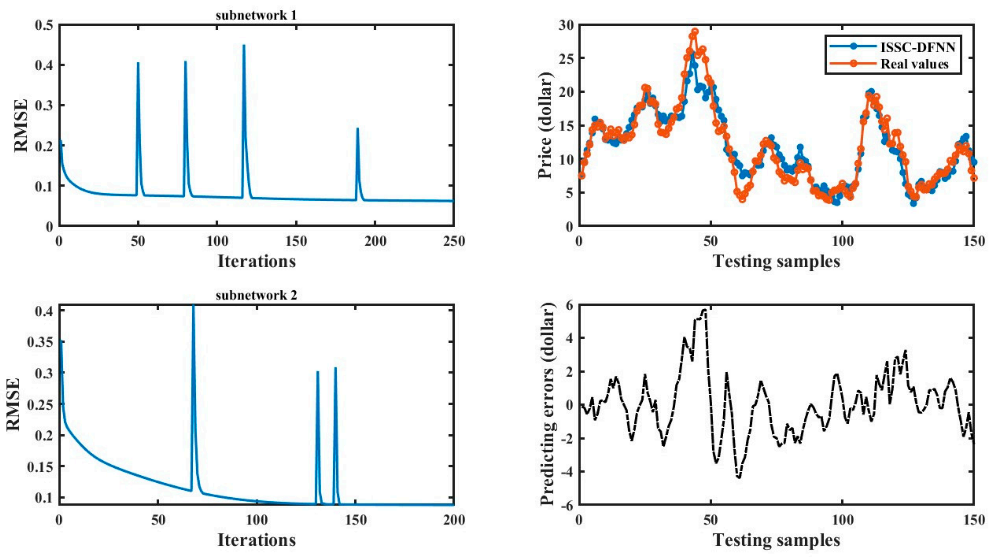

Figure 2. Training and testing results of Boston Housing.

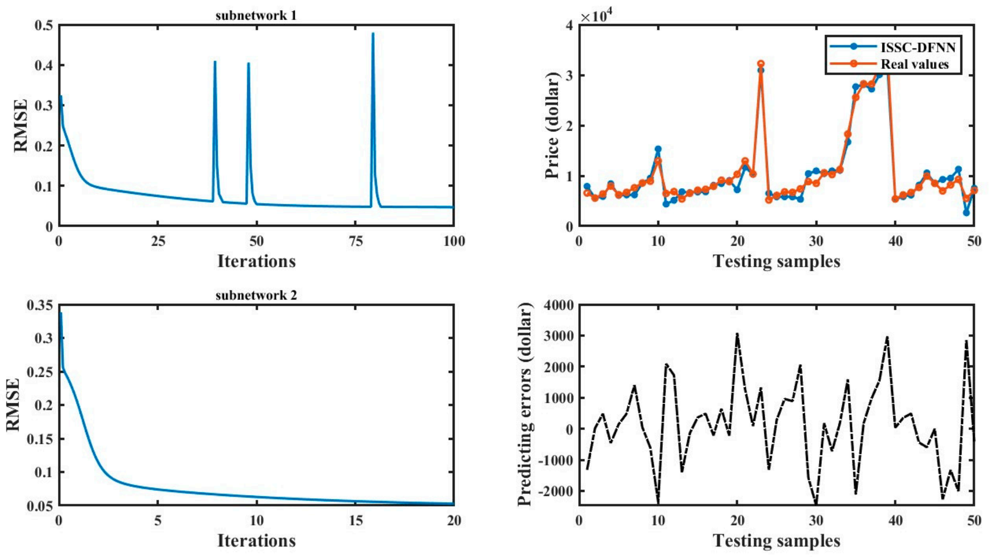

Figure 3. Training and testing results of Auto Price.

C. Prediction results on benchmark problems

Figures 5 illustrate the training and testing results of ISSC-DFNN across different datasets. In each figure, the training RMSE curve gradually converges as the number of iterations increases, with noticeable jumps indicating the addition of hidden nodes within subnetworks to compensate for errors during training. The two subplots on the right display the testing outcomes of ISSC-DFNN, clearly demonstrating its strong approximation ability. Table 5 compares these results with several other models. Besides the evaluation metrics RMSE and MAPE, the number of subnetworks for each model is also reported to highlight structural advantages.

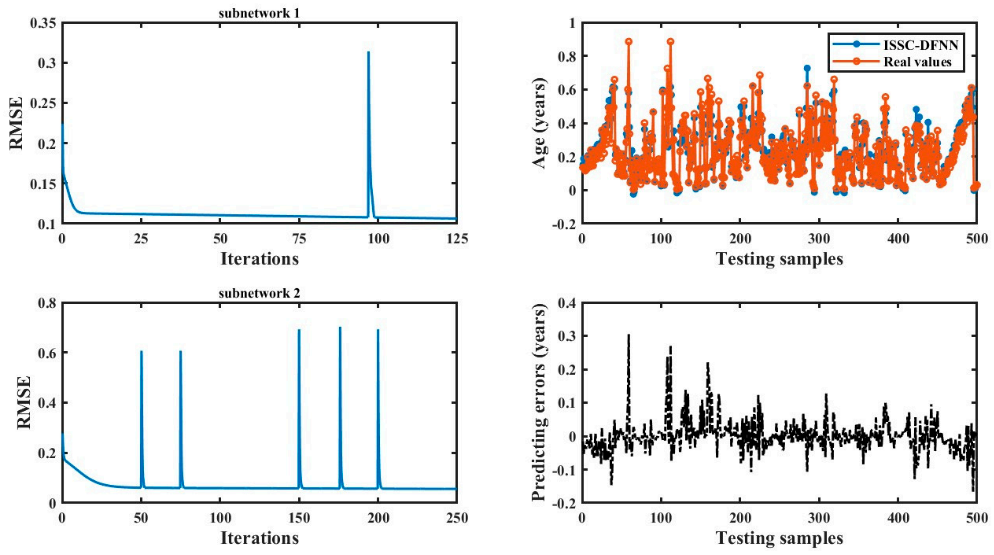

Figure 4. Training and testing results of Abalone.

From Table 6, it is evident that ISSC-DFNN achieves the best performance across multiple metrics, followed by FC-AMNN, both utilize feature clustering methods. In contrast, OSAMNN and TMNN, which are based on sample clustering, exhibit comparatively weaker results. The primary distinction between ISSC-DFNN and FC-AMNN lies in their feature clustering approaches. Furthermore, ISSC-DFNN derives integration weights through task decomposition, making the task decomposition module more tightly coupled with the integration module, thereby enhancing overall model performance.

Table 6. Comparison results with other models on ETP.

Algorithm	Training RMSE	Testing RMSE	Testing APE	Subnetwork
Prop.	**0.0763**	**0.0781**	**0.0306**	**2**
FC-AMNN	0.0802	0.0816	0.0391	2
TMNN	0.0984	0.0996	0.0502	3
OSAMNN	0.0795	0.0828	0.0412	7

The relatively poorer performance of TMNN and OSAMNN can be attributed to several factors, including their sample decomposition strategies, subnetwork architectures, and integration weight calculation. Overall, soft-partition-based task decomposition demonstrates superior generalization ability, as reflected by the lower RMSE and MAPE values when comparing ISSC-DFNN to FC-AMNN. Additionally, ISSC-DFNN overcomes the issue of ineffective subnetwork training caused by insufficient samples in certain subtasks, a problem observed in TMNN and OSAMNN. Notably, ISSC-DFNN features a more compact architecture with fewer subnetworks, implying that it requires a smaller number of subnetworks to achieve the desired accuracy.

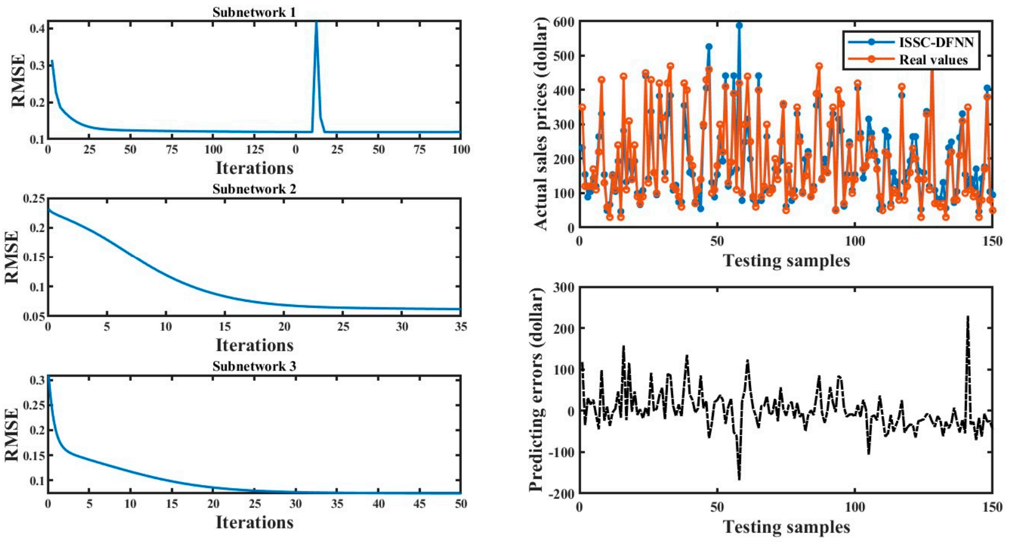

Figure 4. Training and testing results of Resident building.

D. Results on ETP in wastewater treatment prediction

Figure 5 illustrates the testing results of ISSC-DFNN for a single trial on the effluent ETP dataset. As shown, ISSC-DFNN effectively captures the practical data trends, with the fitting curve demonstrating strong approximation capabilities. Table 6 presents the averaged prediction results of ISSC-DFNN alongside several other models over 20 trials for ETP concentration. It is evident from Table 6 that the predictions align well with benchmark outcomes. Similarly, ISSC-DFNN and FC-AMNN, which utilize a feature decomposition approach, outperform other methods on the practical dataset, followed by OSAMNN and TMNN that employ sample decomposition techniques. Notably, both training and testing RMSE values of ISSC-DFNN are lower than those of the competing models, indicating superior generalization performance. Additionally, as reflected in Table 6, ISSC-DFNN maintains a relatively simple architecture with a limited number of subnetworks.

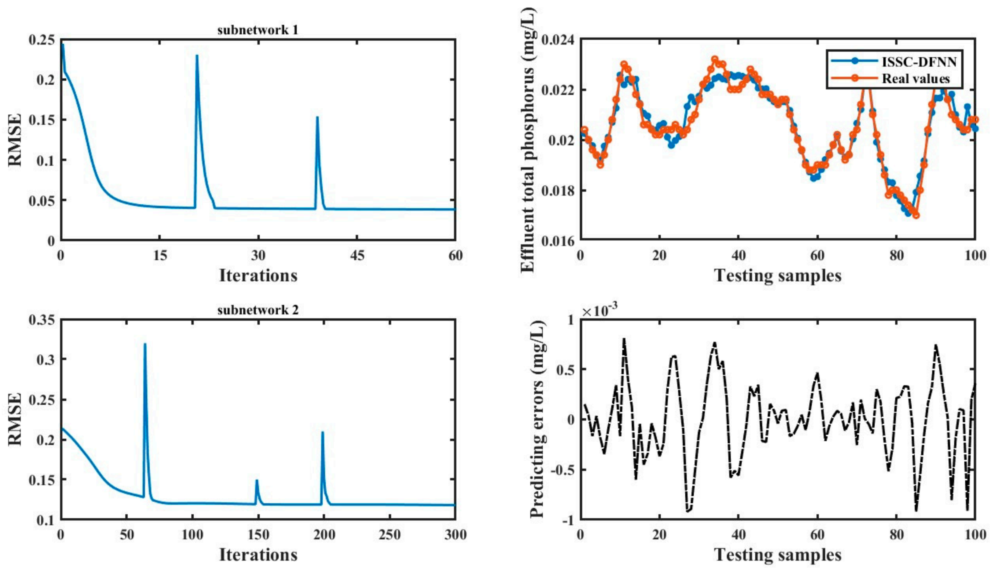

Figure 5. Training and testing results of ETP concentration.

E. Statistical analysis

To further demonstrate the superiority of the proposed model, we conducted a Wilcoxon signed-rank test (Demˇsar, 2006) to compare the performance of ISSC-DFNN with that of several baseline models. The Wilcoxon signed-rank test begins by computing the differences in evaluation metrics between two methods, followed by calculating the rank sums of the positive and negative differences, denoted as R^+ and R^-, respectively. A significant difference is concluded if the resulting probability Pwilconxon value is less than the significance level 0.05. In this study, the testing RMSE is employed as the evaluation metric for all models.

Table 7 presents the results of the Wilcoxon signed-rank test. The experimental results indicate that, regardless of the comparison method, the ISSC-DFNN consistently satisfies the significance criterion of $P_{\text{wilconxon}} < 0.05$. This confirms that the performance of the proposed ISSC-DFNN model is statistically significantly different from that of the other models, thereby further demonstrating its superiority.

Table 7. Wilcoxon's test results of Testing RMSE on all datasets.

Models	R^+	R^-	$P_{\text{wilconxon}}$
Prof. vs TMNN	21	0	0.00741
Prof. vs FC-AMNN	21	0	0.00741
Prof. vs OSAMNN	21	0	0.00741

VI. CONCLUSION

In this study, a distributed neural network framework is proposed, which is built upon the ISSC algorithm. The ISSC algorithm is utilized to perform feature clustering, thereby generating multiple subtasks with partially overlapping feature sets. Each subtask is handled by an RBF neural network trained using a self-organizing approach. Finally, the overall output is obtained by aggregating the predictions of all subnetworks using integration weights derived from the ISSC algorithm. The effectiveness of the proposed ISSC-DFNN has been confirmed by some simulated and experimental results. Tables 5-7 indicate that the proposed ISSC-DFNN achieved better testing RMSE, testing APE, and mean accuracy than the other algorithms. Based on experimental results, the characteristics of ISSC-DFNN are discussed and summarized as follows:

1) When confronted with large-scale datasets, the model is always able to acquire a leaner and denser feature representation, ensuring that each subtask preserves more input information to improve model accuracy.

2) The number of subnetworks can be kept small, even when dealing with datasets of very high dimensionality. Furthermore, based on the self-organizing strategy, each subnet can obtain a relatively compact structure in different subtasks, which is helpful to avoid redundant parameter training and reduce computational complexity.

3) The convergence of the proposed ISSC-DFNN can be maintained, and this proposed ISSC-DFNN with a comprehensive loss function may improve the chance of approximating the global optimization parameters.

Nevertheless, the proposed method still has certain limitations. One notable issue is the reliance on a large number of manually set parameters, including those within the ISSC algorithm as well as the parameters involved in subnetwork training. To address this, future work will focus on developing a more streamlined model with reduced parameter dependency.

ACKNOWLEDGMENT

The authors would like to thank you for reading the manuscript and providing valuable comments. The authors would also like to thank the anonymous reviewers for their valuable comments and suggestions, which helped improve this paper greatly.

References

1. F. J. Lin, I. F. Sun, K. J. Yang, and J. K. Chang, “Recurrent fuzzy neural cerebellar model articulation network fault-tolerant control of six-phase permanent magnet synchronous motor position servo drive,” *IEEE Transactions on Fuzzy Systems*, vol. 24, no. 1, pp. 153-167, Feb. 2016.
2. Y. T. Liu, Y. Y. Lin, S. L. Wu, C. H. Chuang, and C. T. Lin, “Brain dynamics in predicting driving fatigue using a recurrent self-evolving fuzzy neural network,” *IEEE Transactions on Neural Networks and Learning System*, vol. 27, no. 2, pp. 347-360, Feb. 2016.
3. Y. J. Zheng, H. F. Ling, S.Y. Chen, and J. Y. Xue, “ A hybrid neuro-fuzzy network based on differential biogeography-based optimization for online population classification in earthquakes,” *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 4, pp. 1070-1083, Aug. 2015.
4. M. F. Mohammed and C. P. Lim, “An enhanced fuzzy Min–Max neural network for pattern classification,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 3, pp. 417-429, Mar. 2015.
5. R. J. Wai, M. W. Chen, and Y. K. Liu, “Design of adaptive control and fuzzy neural network control for single-stage boost inverter,” *IEEE Transactions on Industrial Electronics*, vol. 62, no. 6, pp. 5434-5445, Sep. 2015.
6. S. Ganjefar and M. Tofighi, “Single-hidden-layer fuzzy recurrent wavelet neural network: Applications to function approximation and system identification,” *Information Sciences*, vol. 294, no. 1, pp. 269-285, Feb. 2015.
7. M. Tofighi, M. Alizadeh, S. Ganjefar, and M. Alizadeh, “Direct adaptive power system stabilizer design using fuzzy wavelet neural network with self-recurrent consequent part,” *Applied Soft Computing*, vol.28, no. 1, pp. 514-526, Mar. 2015.
8. R. Rakkiyappan and P. Balasubramaniam, “On exponential stability results for fuzzy impulsive neural networks,” *Fuzzy Sets and Systems*, vol. 161, no. 13, pp. 1823-1835, Jun. 2010.
9. H. Huang and C. Wu, “Approximation of fuzzy functions by regular fuzzy neural networks,” *Fuzzy Sets and Systems*, vol. 177, no. 1, pp. 60-79, Aug. 2011.
10. R. J. Wai, M. W. Chen, and Y. K. Liu, “Design of adaptive control and fuzzy neural network control for single-stage boost inverter,” *IEEE Transactions on Industrial Electronics*, vol. 62, no. 9, pp. 5434-5445, Sep. 2015.
11. D. Coyle, G. Prasad, and T. M. McGinnity, “Faster self-organizing fuzzy neural network training and a hyperparameter analysis for a brain–computer interface,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 39, no. 6, pp. 1458-1471, Dec. 2009.
12. J. Vanualailai and S. Nakagiri, “Some generalized sufficient convergence criteria for nonlinear continuous neural networks,” *Neural computation*, vol. 17, no. 8, pp. 1820-1835, Aug. 2005.
13. W. Wu, L. Li, J. Yang, and Y. Liu, “A modified gradient-based neuro-fuzzy learning algorithm and its convergence,” *Information Sciences*, vol. 180, no. 9, pp. 1630-1642, May 2010.
14. M. Davanipoor, M. Zekri, and F. Sheikholeslam, “Fuzzy wavelet neural network with an accelerated hybrid learning algorithm,” *IEEE Transactions on Fuzzy Systems*, vol.20, no.3, pp. 463-470, Jun. 2012.

15. C. H. Lee, C. T. Li, and F. Y. Chang, "A species-based improved electromagnetism-like mechanism algorithm for TSK-type interval-valued neural fuzzy system optimization," *Fuzzy Sets and Systems*, vol. 171, no. 1, pp. 22-43, May 2011.
16. C. Ma and L. Jiang, "Some research on Levenberg–Marquardt method for the nonlinear equations," *Applied Mathematics and Computation*, vol. 184, no. 2, pp. 1032-1040, Jan. 2007.
17. M. Kaminski and T. Orlowska-Kowalska, "An online trained neural controller with a fuzzy learning rate of the Levenberg–Marquardt algorithm for speed control of an electrical drive with an elastic joint," *Applied Soft Computing*, vol. 32, no. 1, pp. 509-517, Jul. 2015.
18. Y. K. Yang, T. Y. Sun, C. L. Huo, Y. H. Yu, C. C. Liu, and C. H Tsai, "A novel self-constructing radial basis function neural-fuzzy system," *Applied Soft Computing*, vol. 13, no. 5, pp. 2390-2409, May 2013.
19. N. Ampazis and S. J. Perantonis, "Two highly efficient second-order algorithms for training feedforward networks," *IEEE Transactions on Neural Networks*, vol. 13, no. 5, pp. 1064-1074, Sep. 2002.
20. W. Zhao, K. Li, and G. W. Irwin, "A new gradient descent approach for local learning of fuzzy neural models," *IEEE Transactions on Fuzzy Systems*, vol. 21, no. 1, pp. 30-44, Feb. 2013.
21. C. L. Philip Chen, J. Wang, C. H. Wang, and L. Chen, "A new learning algorithm for a fully connected neuro-fuzzy inference system," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 10, pp. 1741-1757, Oct. 2014.
22. S. T. Tzeng, "Design of fuzzy wavelet neural networks using the GA approach for function approximation and system identification," *Fuzzy sets and systems*, vol. 161, no. 19, pp. 2585-2596, Oct. 2010.
23. R. J. Kuo, S. Y. Hung, and W. C. Cheng, "Application of an optimization artificial immune network and particle swarm optimization-based fuzzy neural network to an RFID-based positioning system," *Information Sciences*, vol. 262, no. 1, pp.78-98, Mar. 2014.
24. M. R. Mashinchi and A. Selamat, "An improvement on genetic-based learning method for fuzzy artificial neural networks," *Applied Soft Computing*, vol. 9, no. 4, pp. 1208-1216, Sep. 2009.
25. C. H. Chen, C. J. Lin, and C. T. Lin, "Nonlinear system control using adaptive neural fuzzy networks based on a modified differential evolution," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 39, no. 4, pp. 459-473, Jul. 2009.
26. H. G. Han, X. L. Wu, and J. F. Qiao, "Nonlinear systems modeling based on self-organizing fuzzy-neural-network with adaptive computation algorithm," *IEEE Transactions on Cybernetics*, vol. 44, no. 4, pp. 554-564, Apr. 2014.
27. A. Miranian and M. Abdollahzade, "Developing a local least-squares support vector machines-based neuro-fuzzy model for nonlinear and chaotic time series prediction," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 2, pp. 207-218, Feb. 2013.
28. H. Yu, P. D. Reiner, T. Xie, T. Bartczak, and B. M. Wilamowski, "An incremental design of radial basis function networks," *IEEE Transactions on* Neural *Networks and Learning Systems*, vol. 25, no. 10, pp. 1793-1803, Oct. 2014.
29. S. Wu, M. J. Er, and Y. Gao, "A fast approach for automatic generation of fuzzy rules by generalized dynamic fuzzy neural networks," *IEEE Transactions on Fuzzy Systems*, vol. 9, no. 4, pp. 578-594, Aug. 2001.
30. S. Wu and M. J. Er, "Dynamic fuzzy neural networks-a novel approach to function approximation," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 30, no. 2, pp. 358-364, Apr. 2000.
31. L. Teslic, B. Hartmann, O. Nelles, and I. Skrjanc, "Nonlinear system identification by Gustafson–Kessel fuzzy clustering and supervised local model network learning for the drug absorption spectra process," *IEEE Transactions on Neural Networks*, vol. 22, no. 12, pp. 1941-1951, Dec. 2011.
32. N. Wang, M. J. Er, and X. Meng, "A fast and accurate online self-organizing scheme for parsimonious fuzzy neural networks," *Neurocomputing*, vol. 72, no. 16, pp. 3818-3829, Oct. 2009.
33. J. J. Rubio, "SOFMLS: online self-organizing fuzzy modified least-squares network," *IEEE Transactions on Fuzzy Systems*, vol. 17, no. 6, pp. 1296-1309, Dec. 2009.
34. H. G. Han and J. F. Qiao, "A self-organizing fuzzy neural network based on a growing-and-pruning algorithm," *IEEE Transactions on Fuzzy Systems*, vol. 18, no. 6, pp. 1129-1143, Dec. 2010.

35. L. Zhang, K. Li, H. He, and G. W. Irwin, "A new discrete-continuous algorithm for radial basis function networks construction," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 11, pp. 1785-1798, Nov. 2013.
36. C. H. Lee and Y. C. Lee, "Nonlinear systems design by a novel fuzzy neural system via hybridization of electromagnetism-like mechanism and particle swarm optimisation algorithms," *Information Sciences*, vol. 186, no. 1, pp. 59-72, Mar. 2012.
37. S. K. Oh, W. D. Kim, W. Pedrycz, and K. Seo. "Fuzzy radial basis function neural networks with information granulation and its parallel genetic optimization," *Fuzzy Sets and Systems*, vol. 237, no. 1, pp. 96-117, Feb. 2014.
38. C. F. Juang, W. Cheng, and C. Liang, "Speedup of learning in interval type-2 neural fuzzy systems through graphic processing units," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 4, pp. 1286-1298, Aug. 2015.
39. W. Zhao, J. Zhang, and K. Li, "An efficient LS-SVM-based method for fuzzy system construction," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 3, pp. 627-643, Jun. 2015.
40. C. F. Juang, Y. Y. Lin, and C. C. Tu, "A recurrent self-evolving fuzzy neural network with local feedbacks and its application to dynamic system processing," *Fuzzy Sets and Systems*, vol. 161, no. 19, pp. 2552-2568, Oct. 2010.
41. N. Wang, M. J. Er, and M. Han, "Dynamic tanker steering control using generalized ellipsoidal- basis-function-based fuzzy neural networks," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 5, pp. 1414-1427, Oct. 2015.
42. S. Tonidandel and J. M. Lebreton, "Relative importance analysis: a useful supplement to regression analysis," *Journal of Business & Psychology*, vol. 26, no. 1, pp. 1-9, Jan. 2011.
43. Y. C. E. Chao, Y. Zhao, L. L. Kupper, and L. A. Nylander-French "Quantifying the relative importance of predictors in multiple linear regression analyses for public health studies," *Journal of Occupational & Environmental Hygiene*, vol. 5, no. 8, pp. 519-29, Aug. 2008.
44. A. Gacek and W. Pedrycz, "Clustering granular data and their characterization with information granules of higher type," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 4, pp. 850-860, Aug. 2015.
45. K. M., Sim, Y. Guo, and B. Shi, "BLGAN: Bayesian learning and genetic algorithm for supporting negotiation with incomplete information," *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, vol. 39, no. 1, pp. 198-211, Feb. 2009.
46. H. Chen, Y. Gong, and X. Hong, "Online modeling with tunable RBF network," *IEEE Transactions on Cybernetics*, vol. 43, no. 3, pp. 935-947, Jun.2013.
47. M. Zubrowska-Sudol and J. Walczak, "Enhancing combined biological nitrogen and phosphorus removal from wastewater by applying mechanically disintegrated excess sludge," *Water Research*, vol. 76, no. 1, pp. 10-18, Jun. 2015.
48. C. Sicard, C. Glen, B. Aubie, D.Wallace, S. Jahanshahi-Anbuhi, K. Pennings, G. T. Daigger, R. Pelton, J. D. Brennan, and C.D.M. Filipe, "Tools for water quality monitoring and mapping using paper-based sensors and cell phones," *Water Research*, vol. 70, no. 1, pp. 360-369, Mar. 2015.
49. M. K. Winkler, K. F. Ettwig, T. P. Vannecke, K. Stultiens, A. Bogdan, B. Kartal, and E. I. P. Volcke, "Modelling simultaneous anaerobic methane and ammonium removal in a granular sludge reactor," *Water Research*, vol. 73, no. 15, pp. 323-331, Apr. 2015.
50. W. Gujer, "Nitrification and me-A subjective review," *Water Research*, vol. 44, no. 1, pp. 1-19, Jan. 2010.
51. D. Kaelin, R. Manser, L. Rieger, J. Eugster, K. Rottermann, and H. Siegrist, "Extension of ASM3 for two-step nitrification and denitrification and its calibration and validation with batch tests and pilot scale data," *Water Research*, vol. 43, no. 6, pp. 1680-1692, Apr. 2009.