A Deep Nonconnectionist Learning Framework for Industrial Data Modeling

Yongxuan Chen and Dianhui Wang, *Senior Member, IEEE*

***Abstract*—Despite the extensive applications of deep neural networks in data modeling filed, there are still some obvious deficiencies for the implementation in modern industrial cases. There are mainly reflected in the following aspects: first, the architectures are difficult to configure; second, the modeling process is time-consuming; third, the training procedure easily falls into the local optimum situation. To overcome these problems, exploring the nonconnectionist learning model has become a popular topic recently. This article proposes a deep nonconnectionist learning model based on kernel principal component regression (KPCR), which is referred to as stacked KPCR (SKPCR). By stacking multiple KPCR modules, a multilayer learning model is constructed by adopting hierarchical feature extraction. In SKPCR, the model structure is determined incrementally and there is only one parameter needed to be configured for each layer. Furthermore, an enhanced learning strategy is designed for alleviating the information loss problem in the training process. An actual industrial case is used to validate the effectiveness, including the prediction performance and modeling efficiency, of our proposed method.**

***Index Terms*—Data-driven modeling, deep learning, industrial processes, kernel principal component regression (KPCR), quality prediction, soft sensor.**

I. Introduction

MODELING technology plays an indispensable role for industrial intelligence. With the rapid progress of distributed control systems (DCSs), the acquisition of process data becomes much more convenient. These process data reflect the operation situation at the present moment, which is significant for process control and monitoring. How to utilize the huge amount of industrial data are crucial and valuable [1], [2], [3]. In the past few decades, multivariate statistical analysis methods, such as partial least squares (PLS) [4] and principal component analysis (PCA) [5] are extensively applied in the industrial processes for fault monitoring and quality prediction. Fundamentally, these methods belong to linear learning model, which does not accord with the complex situations of industrial processes. Hence, the nonlinear forms of these models are developed to better describe the processes features. One of these extensions is called the kernel method, which applies the kernel trick to execute the PLS or PCA in the high-dimension feature space. The distinguished feature about kernel-based methods is that the complex definition of high-dimension mapping is avoided, which makes these models easy to construct and is very suitable for industrial applications. Therefore, the kernel versions of PLS and PCA are extensively implemented for industrial data modeling [6], [7], [8].

In recent years, the deep learning technology has demonstrated its powerful capability for data analysis [9], [10]. The fundamental idea of deep learning is to stack multiple basic nonlinear modules, then the very complicated functional expressions can be expressed. Deep neural network (DNN) is the most representative method of the multilayer architecture learners. Composed of several nonlinear neurons, each layer of DNN can slightly increase the representation ability of the model. By stacking enough layers, a very complicated function is expected to be learned by a DNN. With this excellent characteristic, many types of networks are designed for different application situation, such as stacked autoencoder (SAE) [11], recurrent neural networks [12], and convolutional neural networks [13].

In industrial data modeling field, researchers have also introduced and improved the deep networks for enhancing the industrial intelligence [14], [15], [16], [17]. Although these network-based models have shown strong abilities for industrial data modeling, there are still some limits for the real application for industrial processes. First, the architectures of deep networks usually need to be predefined before the training process. That is very difficult without the prior knowledge. Second, the training process of a deep network is time-consuming. The backpropagation algorithm is usually adopted for adjusting the model parameters, which is executed iteratively. For a large-scale network, the computational cost is sometimes unacceptable for industrial application cases. Third, the combination of training parameters is hard to determine optimally. The training parameters for a deep network include batch size, the largest iteration number and learning rate. Each of these parameters is crucial for the modeling performances, and it is tough to find the appropriate combination for these parameters. Especially for a deep

Manuscript received 26 January 2024; revised 21 March 2024; accepted 18 May 2024. Date of publication 5 June 2024; date of current version 28 June 2024. This work was supported by the National Key Research and Development Program of China under Grant 2018AAA0100304. Recommended by Lead Guest Editor Yuemin Ding and Guest Editor Fei Pan. *(Corresponding author: Dianhui Wang.)*

Yongxuan Chen is with the School of Low-Carbon Energy and Power Engineering, Xuzhou 221116, China, and also with the Artificial Intelligence Research Institute, China University of Mining and Technology, Xuzhou 221116, China (e-mail: tb23130011p41@cumt.edu.cn).

Dianhui Wang is with the Artificial Intelligence Research Institute, China University of Mining and Technology, Xuzhou 221116, China, and also with the State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang 110819, China (e-mail: dh.wang@deepscn.com).

Digital Object Identifier 10.1109/JSAS.2024.3404416

network, the modeling accuracy is very sensitive to the training parameters. In another word, the deeper the network is, the harder the parameter combination is to be configured. Fourth, training a deep network needs a huge number of samples. If the number of training samples is too small, it is with great chance to fall into the overfitting problem. Fifth, the prediction outcomes of a deep network may not be global optimal. This issue is mainly attributed to the mechanism of backpropagation algorithm. The core computation of backpropagation algorithm is the derivation operation, which is easily fallen into local optimality when the structure of deep network is too complicated. All those problems limit the further applications of deep networks in the industrial modeling field.

To alleviate the aforementioned problems, researchers focus on developing the nonconnectionist deep learning model with more concise structures. Mitchell and Sheppard [18] first discussed the deep nonconnectionist architectures using PCA for image classification experiments. Chen and Deng [20] utilized kernel PLS as the basic learning module for constructing an efficient deep learning model. To alleviate the information loss problem of [20], Chen and Wang [19] proposed an enhanced deep kernel PLS by using identity-mapping structure. Kong and Ge [21] constructed a deep model based on independent component analysis and PCA for process monitoring. These nonconnectionist deep models have shown their superiorities over the traditional deep networks in terms of modeling accuracy and efficiency, which becomes one of the most popular methods for industrial data analysis.

The essence of these novel deep models is to simply stack multiple basic modules, which is effective in terms of increasing the model complexity. However, the basic learning modules, such as PCA and PLS, are fundamentally the dimension-reduction models, and it is conceivable that, with the depth continuously increases, the information loss of these nonconnectionist deep models is expected to raise, which is harmful for modeling precision. Therefore, the construction of nonconnectionist deep learning models still has huge rooms for improvement.

This article extends the family of nonconnectionist deep learning model by exploring the deep learning version of kernel principal component regression (KPCR), which is referred to as stacked KPCR (SKPCR). KPCR is a nonlinear multivariate statistical method, which is commonly used for data modeling analysis. By introducing the kernel technique into the traditional PCR method, the principal components extracted by KPCR contain nonlinear information of data and are suitable for nonlinear process modeling application. However, KPCR belongs to the shallow learning machine and may not perform well in the cases where strong nonlinearities exist. Distilling the concept of deep learning, traditional KPCR is deepened through constructing a multilayer architecture in this article, where the hierarchical feature extraction is executed. Furthermore, to alleviate the information loss problem, an enhanced learning strategy is designed to compensate for the dimension-reduction process of KPCR. Attributed to the simplicity of KPCR modeling procedures, there is only one parameter needed to be configured for each layer. As a result, the computational cost of SKPCR is much lower than that of deep networks. A real-world industrial case is used to validate the effectiveness of SKPCR.

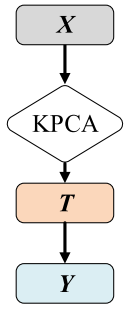

Fig. 1. Schematic of KPCR.

The contributions of this work can be clarified as follows.

1) A novel deep nonconnectionist learning model is proposed based on KPCR, where the hierarchical feature extraction is employed for mining deep data features.
2) An enhanced learning strategy is designed to alleviate the information loss problem during the model training process of the proposed method.
3) A real-world industrial case is used to evaluate the effectiveness of our proposed method, where the proposed method is compared to the state-of-art deep learning methods.

The rest this article is organized as follows. Section II briefly reviews the basic definition of KPCR. Section III details the proposed SKPCR method. Section IV gives an actual industrial application to demonstrate the effectiveness of our proposed method. Finally, Section V concludes this article.

II. Kernel Principal Component Regression

The modeling process of KPCR is composed of two steps: principal components extraction by KPCA and regression model construction using principal components [21]. The schematic of KPCR is demonstrated in Fig. 1.

Given normalized input data $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]$ with n samples and m variables, and output data $\boldsymbol{y} = [y_1, y_2, \ldots, y_n]$ with n samples. A nonlinear mapping $\boldsymbol{\Phi}$ is performed to map the raw data onto the high-dimension feature space $\mathcal{F}$

$$\boldsymbol{x}_i \in \mathbb{R}^m \rightarrow \boldsymbol{\Phi}\left(\boldsymbol{x}_i\right) \in \mathcal{F}. \tag{1}$$

It is believed that, in the feature space $\mathcal{F}$, the functional relation between $\boldsymbol{\Phi}(\boldsymbol{X})$ and $\boldsymbol{y}$ is more likely to have the linear form. Therefore, PCA method can be employed for extracting the principal components.

The covariance matrix $\boldsymbol{C}$ of the mapped data $\boldsymbol{\Phi}(\boldsymbol{X})$ in feature space is calculated as

$$\boldsymbol{C} = \frac{1}{n}\boldsymbol{\Phi}\left(\boldsymbol{X}\right)\boldsymbol{\Phi}(\boldsymbol{X})^{\mathrm{T}} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{\Phi}\left(\boldsymbol{x}_i\right)\boldsymbol{\Phi}(\boldsymbol{x}_i)^{\mathrm{T}}. \tag{2}$$

Then, the next step is to find the eigenvalues and eigenvectors of $\boldsymbol{C}$, which is formulated as $\boldsymbol{C}\boldsymbol{p} = \lambda\boldsymbol{p}$. This problem is

equivalent to

$$\sum_{i=1}^{n} \boldsymbol{\Phi}\left(\boldsymbol{x}_i\right) \boldsymbol{\Phi}\left(\boldsymbol{x}_i\right)^{\mathrm{T}} \boldsymbol{p} \;=\; n\lambda \boldsymbol{p}. \tag{3}$$

Since the concrete form of $\boldsymbol{\Phi}$ is unknown, the aforementioned equation is unable to be solved. Here, the kernel trick [7] is adopted to complete the calculation without the awareness of $\boldsymbol{\Phi}$. The kernel trick indicates that, the dot product of two vectors in the high-dimension space can be computed in the original data space through kernel function.

Define the kernel function as $k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \boldsymbol{\Phi}\left(\boldsymbol{x}_i\right)^{\mathrm{T}} \boldsymbol{\Phi}\left(\boldsymbol{x}_j\right)$, where $k(.,.)$ represents the kernel function. There are many types of kernel functions, the Gaussian kernel function is used in our work as

$$k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \exp\left(-\frac{\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2}{c}\right) \tag{4}$$

where c is the kernel parameter needed to be configured artificially.

Here, let us refocus on (3) and rewrite it as

$$\boldsymbol{p} \;=\; \frac{1}{n\lambda} \sum_{i=1}^{n} \boldsymbol{\Phi}\left(\boldsymbol{x}_i\right) \boldsymbol{\Phi}\left(\boldsymbol{x}_i\right)^{\mathrm{T}} \boldsymbol{p} \tag{5}$$

where $\boldsymbol{\Phi}\left(\boldsymbol{x}_i\right)^{\mathrm{T}} \boldsymbol{p}$ is a scalar. That means eigenvector $\boldsymbol{p}$ can be formulated by the linear combination of $\boldsymbol{\Phi}\left(\boldsymbol{x}_i\right)$ as

$$\boldsymbol{p} \;=\; \sum_{i=1}^{n} \alpha_i \boldsymbol{\Phi}\left(\boldsymbol{x}_i\right) \;=\; \boldsymbol{\Phi}\left(\boldsymbol{X}\right) \boldsymbol{\alpha} \tag{6}$$

where $\boldsymbol{\alpha} \;=\; [\alpha_1, \alpha_2, \ldots, \alpha_n]$.

Combinining (3) and (6), we can obtain

$$\boldsymbol{\Phi}\left(\boldsymbol{X}\right) \boldsymbol{\Phi}\left(\boldsymbol{X}\right)^{\mathrm{T}} \boldsymbol{\Phi}\left(\boldsymbol{X}\right) \boldsymbol{\alpha} \;=\; n\lambda \boldsymbol{\Phi}\left(\boldsymbol{X}\right) \boldsymbol{\alpha} \tag{7}$$

and premultiply $\boldsymbol{\Phi}\left(\boldsymbol{X}\right)^{\mathrm{T}}$, which leads to

$$\boldsymbol{\Phi}\left(\boldsymbol{X}\right)^{\mathrm{T}} \boldsymbol{\Phi}\left(\boldsymbol{X}\right) \boldsymbol{\Phi}\left(\boldsymbol{X}\right)^{\mathrm{T}} \boldsymbol{\Phi}\left(\boldsymbol{X}\right) \boldsymbol{\alpha} \;=\; n\lambda \boldsymbol{\Phi}\left(\boldsymbol{X}\right)^{\mathrm{T}} \boldsymbol{\Phi}\left(\boldsymbol{X}\right) \boldsymbol{\alpha}. \tag{8}$$

Define the Gram matrix $\boldsymbol{K} \;=\; \boldsymbol{\Phi}\left(\boldsymbol{X}\right)^{\mathrm{T}} \boldsymbol{\Phi}\left(\boldsymbol{X}\right)$, and the element of $\boldsymbol{K}$ is calculated by the kernel function as

$$\boldsymbol{K}\;(i,j) \;=\; k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right). \tag{9}$$

Therefore, (8) is reformulated as

$$\boldsymbol{K}\boldsymbol{K}\boldsymbol{\alpha} \;=\; n\lambda \boldsymbol{K}\boldsymbol{\alpha} \tag{10}$$

which is equivalent to

$$\boldsymbol{K}\boldsymbol{\alpha} \;=\; n\lambda \boldsymbol{\alpha}. \tag{11}$$

Hence, $\boldsymbol{\alpha}$ represents the eigenvectors of Gram matrix $\boldsymbol{K}$. Usually, $\boldsymbol{\alpha}$ needs to be normalized as $\|\boldsymbol{\alpha}\| = \frac{1}{n\lambda}$.

So far, the original problem is transformed into the eigenvalue decomposition of Gram matrix $\boldsymbol{K}$, which can be calculated by data matrix $\boldsymbol{X}$. Reformulate $\boldsymbol{K}$ through eigenvalue decomposition as

$$\boldsymbol{K}\boldsymbol{A} \;=\; \boldsymbol{\Delta}\boldsymbol{A} \tag{12}$$

where $\boldsymbol{\Delta}$ and $\boldsymbol{A}$ represent the eigenvalue matrix and eigenvector matrix in descending order by the eigenvalue, respectively. For the purpose of dimension reduction, only the first d columns of $\boldsymbol{A}$ are used as the projection matrix. The value of d can be set by the contribution of corresponding eigenvalues.

Therefore, the principal components $\boldsymbol{T}$ of KPCA are calculated by

$$\boldsymbol{T} \;=\; \boldsymbol{K}\boldsymbol{A}_d \tag{13}$$

where $\boldsymbol{A}_d$ is the matrix consisted of the first d columns of $\boldsymbol{A}$, and the linear regression model can be established between $\boldsymbol{T}$ and $\boldsymbol{y}$ as

$$\boldsymbol{y} \;= \boldsymbol{T}\left(\boldsymbol{T}^{\mathrm{T}}\boldsymbol{T}\right)^{-1} \boldsymbol{T}^{\mathrm{T}}\boldsymbol{y} + \hat{\boldsymbol{y}} \tag{14}$$

where $\hat{\boldsymbol{y}}$ is the residual vector.

For the unknown query data $\boldsymbol{X}_t$, the Gram matrix $\boldsymbol{K}_t$ is firstly computed by $\boldsymbol{X}_t$ and $\boldsymbol{X}$ using (9). Then, the principal components $\boldsymbol{T}_t$ of query data are calculated by

$$\boldsymbol{T}_t = \boldsymbol{K}_t\, \boldsymbol{A}_d \tag{15}$$

and the prediction values for $\boldsymbol{X}_t$ are

$$\boldsymbol{y}_t = \boldsymbol{T}_t\left(\boldsymbol{T}^{\mathrm{T}}\boldsymbol{T}\right)^{-1} \boldsymbol{T}^{\mathrm{T}}\boldsymbol{y} + \hat{\boldsymbol{y}}_t. \tag{16}$$

III. Stacked KPCR

KPCR is essentially the shallow learning machine, which cannot capture the intrinsic data feature information. To deepen the traditional KPCR, a multilayer KPCR model assisted by the enhanced learning strategy is proposed in this section. This hierarchical structure strengthens the nonlinear expression ability by reconstructing the kernel mapping form at each layer. The proposed model is easier to build compared with the deep networks and has stronger learning ability compared with the shallow learning machines.

A. Multilayer KPCR

The core of deep learning lies on the multiple representation-learning. By composing several simple but nonlinear modules, very complicated functions can be expressed by the deep learning model [10]. Therefore, the purpose of our work is to first establish the multilayer architecture of KPCR.

Given normalized data $\boldsymbol{X}$, they are first processed by KPCA to extract the d_1-dimension principal components $\boldsymbol{T}_1 = [\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_{d_1}]$ by (13), where the number of d_1 can be determined by the setting contribution rate.

Then, the components $\boldsymbol{T}_1$ from the 1st layer are utilized to construct the Gram matrix for the second layer as

$$\boldsymbol{K}_2\;(i,j) = k\;(\boldsymbol{t}_i, \boldsymbol{t}_j) = \exp\left(-\frac{\left\|\boldsymbol{t}_i - \boldsymbol{t}_j\right\|^2}{c_2}\right) \tag{17}$$

where $i, j \;=\; 1,\; 2, \ldots, d_1$. The eigenvalue decomposition procedures are employed again on $\boldsymbol{K}_2$ for extracting the principal components $\boldsymbol{T}_2$ at the second layer

$$\boldsymbol{T}_2 = \boldsymbol{K}_2\, \boldsymbol{A}_{d_2} \tag{18}$$

where $\boldsymbol{A}_{d2}$ is the first d_2 columns of eigenvector matrix of $\boldsymbol{K}_2$.

For the lth layer, where $3 \leq l \leq L$(L is the depth of model), the deep features $\boldsymbol{T}_l$ are computed by

$$\boldsymbol{T}_l = \boldsymbol{K}_l \, \boldsymbol{A}_{d_l} \tag{19}$$

where $\boldsymbol{K}_l$ is the Gram matrix composed by the principal components $\boldsymbol{T}_{l-1}$ from the previous $l-1$ layer

$$\boldsymbol{K}_l \ (i,j) = \ k \ (\boldsymbol{t}_{l-1,i}, \boldsymbol{t}_{l-1,j}) = \exp\left(-\frac{\|\boldsymbol{t}_{l-1,i} - \boldsymbol{t}_{l-1,j}\|^2}{c_l}\right). \tag{20}$$

The deep features $\boldsymbol{T}_L$ are used for the final regression model building by (14)

$$\boldsymbol{y} \ = \boldsymbol{T}_L \left(\boldsymbol{T}_L{}^{\mathrm{T}} \boldsymbol{T}_L\right)^{-1} \boldsymbol{T}_L{}^{\mathrm{T}} \boldsymbol{y} + \hat{\boldsymbol{y}}. \tag{21}$$

For query data $\boldsymbol{X}_t$, they are needed to be transformed into the deep features successively for giving the estimation values. The testing Gram matrix at the first layer is computed by raw data $\boldsymbol{X}$ and $\boldsymbol{X}_t$

$$\boldsymbol{K}_{t,1} \ (i,j) = \ k \ (\boldsymbol{x}_i, \boldsymbol{x}_{t,j}) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_{t,j}\|^2}{c_1}\right) \tag{22}$$

and the corresponding data features can be calculated as

$$\boldsymbol{T}_{t,1} = \boldsymbol{K}_{t,1} \, \boldsymbol{A}_{d_1} \tag{23}$$

where the projection matrix $\boldsymbol{A}_{d_1}$ has already been computed in the modeling stage. For lth layer, where $2 \leq l \leq L$(L is the depth of model), the testing deep features are

$$\boldsymbol{T}_{t,l} = \boldsymbol{K}_{t,l} \, \boldsymbol{A}_{d_l} \tag{24}$$

where $\boldsymbol{K}_{t,l}$ is the Gram matrix composed by features from the previous layer $\boldsymbol{T}_{l-1}$ and $\boldsymbol{T}_{t,l-1}$.

The final prediction values for $\boldsymbol{X}_t$ are

$$\boldsymbol{y}_t = \boldsymbol{T}_{t,L} \left(\boldsymbol{T}_L{}^{\mathrm{T}} \boldsymbol{T}_L\right)^{-1} \boldsymbol{T}_L{}^{\mathrm{T}} \boldsymbol{y} + \hat{\boldsymbol{y}}_t. \tag{25}$$

The aforementioned modeling procedures transform KPCR into a deep learning model by adopting the hierarchical learning mechanism, where highly abstract features $\boldsymbol{T}_{t,L}$ and $\boldsymbol{T}_L$ are used for the final regression analysis.

B. Stacked KPCR

In multilayer KPCR, the raw data are successively transformed into the deep features for regression model building. However, this may exist a problem: the dimension reduction operations of KPCA are expected to cause the information loss. The features extracted by KPCA represent the major information of the unprocessed data, which is beneficial for reducing the noise and declining the computational complexity. However, this operation also indicates that a small portion of the data information is discarded. With the depth of multilayer KPCR continuously increases, the accumulation of information loss is accordingly raised. That is harmful for reserving the useful regression information at the modeling stage.

To alleviate this problem, one straightforward way is to enrich the data information at each layer. In this section, an enhanced learning strategy is designed for better training the multilayer KPCR, which is referred to as SKPCR. In SKPCR, the original raw data $\boldsymbol{X}$ participate in the feature extraction process at each layer.

At the first layer of SKPCR, raw data $\boldsymbol{X}$ are processed by KPCA to obtain the data features $\boldsymbol{T}_1 = [\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_{d_1}]$ by (13). For the second layer, $\boldsymbol{T}_1$ are not directly used for establishing the Gram matrix. Instead, an enhanced input matrix is built as

$$\boldsymbol{E}_1 = \left[\boldsymbol{T}_1, \boldsymbol{X}^{\mathrm{T}}\right] \ = [\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_{d_1}, \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \ \boldsymbol{x}_m] \tag{26}$$

where $\boldsymbol{x}_p (p = 1, 2, \ldots, m)$ represents the pth variables of $\boldsymbol{X}$. Then, the enhanced inputs $\boldsymbol{E}_1 = [\boldsymbol{e}_{1,1}, \boldsymbol{e}_{1,2}, \ldots, \boldsymbol{e}_{1,d_1+m}]$ are used to construct the Gram matrix for the next layer

$$\boldsymbol{K}_2 \ (i,j) = \ k \ (\boldsymbol{e}_{1,i}, \boldsymbol{e}_{1,j}) = \exp\left(-\frac{\|\boldsymbol{e}_{1,i} - \boldsymbol{e}_{1,j}\|^2}{c_2}\right). \tag{27}$$

According to KPCA, the data features at the second layer are

$$\boldsymbol{T}_2 = \boldsymbol{K}_2 \, \boldsymbol{A}_{d_2} \tag{28}$$

and for the lth layer, where $3 \leq l \leq L$(L is the depth of model), the data features are

$$\boldsymbol{T}_l = \boldsymbol{K}_l \, \boldsymbol{A}_{d_l} \tag{29}$$

where $\boldsymbol{K}_l$ is the Gram matrix composed by the enhanced inputs $\boldsymbol{E}_{l-1}$ from the previous $l-1$ layer

$$\boldsymbol{E}_{l-1} = \left[\boldsymbol{T}_{l-1}, \boldsymbol{X}^{\mathrm{T}}\right] \ = [\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_{d_{l-1}}, \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \ \boldsymbol{x}_m] \tag{30}$$

and the prediction values are produced using deep features $\boldsymbol{T}_l$ by (21).

For query data $\boldsymbol{X}_t$, they are also transformed into the deep features through data augmentation. At the first layer, data features $\boldsymbol{T}_{t,1}$ are obtained by (22) and (23). Then, the enhanced inputs for the second layer are constructed by

$$\boldsymbol{E}_{t,1} = \left[\boldsymbol{T}_{t,1}, \boldsymbol{X}_t{}^{\mathrm{T}}\right] = [\boldsymbol{t}_{t,1}, \boldsymbol{t}_{t,2}, \ldots, \boldsymbol{t}_{t,d_1}, \boldsymbol{x}_{t,1}, \boldsymbol{x}_{t,2}, \ldots, \ \boldsymbol{x}_{t,m}] \tag{31}$$

where $\boldsymbol{x}_{t,p} (p = 1, 2, \ldots, m)$ represents the pth variables of $\boldsymbol{X}_t$. Accordingly, the enhanced inputs feeding to the lth layer, where $2 \leq l \leq L$(L is the depth of model), are $\boldsymbol{E}_{t,l-1} = [\boldsymbol{T}_{t,l-1}, \boldsymbol{X}_t{}^{\mathrm{T}}]$. At the lth layer, deep features of $\boldsymbol{X}_t$ are formulated as

$$\boldsymbol{T}_{t,l} = \boldsymbol{K}_{t,l} \, \boldsymbol{A}_{d_l} \tag{32}$$

where $\boldsymbol{K}_{t,l}$ is the Gram matrix of enhanced inputs $\boldsymbol{E}_{l-1}$ and $\boldsymbol{E}_{t,l-1}$.

After all L layers' feature extraction procedures are completed, deep features $\boldsymbol{T}_{t,L}$ are used for the prediction by (25). The schematic of SKPCR is given in Fig. 2.

Remark 1: Compared to multilayer KPCR, there is only one extra operation of SKPCR: the inputs feeding to each layer are augmented by the original data matrix $\boldsymbol{X}$ and $\boldsymbol{X}_t$ (except for the first layer). The purpose of this design is to compensate for the information loss problem caused by the dimension reduction operations of KPCA. The principal components extract by KPCA represent the major information of original data, which means a small part of the data information is discarded. This small part of the data information is expected to be the noise or irrelevant information. However, with the accumulation of information loss by multiple layers, deep features extracted by

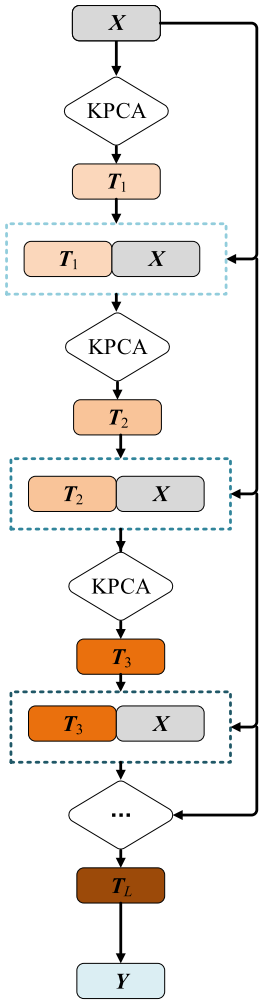

Fig. 2. Schematic of stacked KPCR (SKPCR).

multilayer KPCR may have lost major characteristics of original data and not be the appropriate representation for regression analysis. The design of SKPCR alleviates this problem by using original data ($\boldsymbol{X}$ and $\boldsymbol{X}_t$) to construct the enhanced inputs for each layer. In this way, the inputs for each layer contain the information both from previous layer and the original data.

The augmentation operation of SKPCR at each layer does not increase the computational cost greatly due to the mechanism of kernel technique. From (4) and (9), it is clear that the scale of Gram matrix is related to the number of data samples, and the enhanced inputs do not change the scale of Gram matrix at each layer.

Remark 2: There is only one parameter needed to be configured at each layer of SKPCR, which is the kernel parameter c_l at lth layer. In our work, the number of principal components is automatically determined by the setting contribution rate, and the kernel parameter is determined by exhaustive search using validation data. In this way, the selected parameter is global optimum under certain searching range.

Therefore, the aim of training a SKPCR model is to determine the suitable kernel parameter for each layer. Accordingly, the training process for SKPCR is conducted incrementally. Beginning from the conventional KPCR, extra layer is added to the current model successively until validation error no longer descends.

TABLE I
MODELING PROCEDURES OF SKPCR

Training Phase:

>Collect historical data and divide them into training dataset $\{\boldsymbol{X}, \boldsymbol{y}\}$ and validation dataset $\{\boldsymbol{X}_v, \boldsymbol{y}_v\}$.
>Set contribution rate and searching range of kernel parameter.
>Compute 1st layer's data feature $\boldsymbol{T}_1$ by KPCA.
>Set $l = 2$;
1) Construct enhanced input data $[\boldsymbol{T}_{l-1}, \boldsymbol{X}^{\mathrm{T}}]$.
2) Compute lth layer's data feature $\boldsymbol{T}_l$ by KPCA under the objective function (33).
3) If $e_v^{l)} < e_v^{l-1)}$, set $l = l + 1$ and return to 1).
End
>Set model depth $L = l - 1$;
>Obtain modeling prediction by (21).

Application Phase:

>Obtain query data $\boldsymbol{X}_t$.
> Compute 1st layer's data feature $\boldsymbol{T}_{t,1}$ by (23).
> For $l = 2{:}\,L$;
1) Construct enhanced input data $\left[\boldsymbol{T}_{t,l-1}, \boldsymbol{X}_t{}^{\mathrm{T}}\right]$.
2) Compute lth layer's data feature $\boldsymbol{T}_{t,l}$ by (32).
End
> Obtain deep features $\boldsymbol{T}_{t,L}$.
>Obtain testing prediction with by (25).

C. Model Training Strategy

The objective of training a SKPCR model is to properly determine the model parameters. There are two parameters of KPCA method: the number of principal components and the kernel parameter. In our work, the number of principal components is automatically determined by the global-setting contribution rate, and the kernel parameter for each layer is determined by exhaustive search using validation data.

Let $\{\boldsymbol{X}_v, \boldsymbol{y}_v\}$ represents the validation dataset, the validation error $e_v^{l)}$ at lth layer is computed by

$$e_v^{l)} = \frac{1}{n_v} \sum_{a}^{n_v} \sqrt{\left(f_l\left(\boldsymbol{x}_{v,a}\right) - y_{v,a}\right)^2} \tag{33}$$

where n_v is the number of validation dataset, $f_l(\cdot)$ denotes the current model with l layers, $\boldsymbol{x}_{v,a}$ and $y_{v,a}$ represent the ath sample of $\boldsymbol{X}_v$ and $\boldsymbol{y}_v$.

The kernel parameter c_l selected for this layer is to produce the minimum validation error $e_v^{l)}$ under searching range $1:1:c_{\max}$. The value of $c_{\max}$ is configured before the training process. Since all potential candidates are evaluated in this stage, it is guaranteed that the selected parameter is global optimization under the searching range $1:1:c_{\max}$.

D. Modeling Procedures of SKPCR

The modeling procedures of SKPCR can be divided into two steps: offline training stage and online application stage (the

detailed procedures can be found in Table I). At the training stage, SKPCR model is built based on historical data. At the application stage, the trained model is used to give the prediction values for upcoming unknown data. The details are illustrated as follows.

Offline modeling stage:

1) Obtain historical data and separate them into training dataset $\{\boldsymbol{X},\ \boldsymbol{y}\}$ and validation dataset $\{\boldsymbol{X}_v, \boldsymbol{y}_v\}$. Standardize them by the variance and mean of training dataset.
2) Set contribution rate and searching range $1:1:c_{\max}$ for kernel parameter.
3) Transform $\boldsymbol{X}$ into deep features $\boldsymbol{T}_L$ by (29) and (30), and search the best kernel parameter for each layer by (33), which can reach the minimum prediction error under validation dataset.
4) Set model length $L = l$ while $e_v^{l)} \leq e_v^{l+1)}$, the training process is completed.

Online application stage:

1) Obtain query data $\boldsymbol{X}_t$ and standardize them by the variance and mean of training dataset.
2) Transform $\boldsymbol{X}_t$ into deep features $\boldsymbol{T}_{t,L}$ successively by (32).
3) Give the prediction by (25).

IV. Case Study

This section uses an actual industrial case to verify the effectiveness of our proposed method, including the comparisons between several state-of-art modeling methods. Two common indicators, root-mean-squares error (RMSE) and R^2 (coefficient of determination), are used for quantifying the modeling performances. The formulations of RMSE and R^2 are

$$\mathrm{RMSE} = \sqrt{\frac{\sum (\boldsymbol{y}_t - \hat{\boldsymbol{y}}_t)^2}{N}} \tag{34}$$

$$\mathrm{R}^2 = 1 - \frac{\sum (\boldsymbol{y}_t - \hat{\boldsymbol{y}}_t)^2}{\sum (\boldsymbol{y}_t - \bar{\boldsymbol{y}}_t)^2} \tag{35}$$

where N is the number samples, y_t are the real values of query samples and $\hat{\boldsymbol{y}}_t$ are the prediction values produced by a learning model.

A. Process Description

In the petroleum refining industry, debutanizer column is an important unit for separating naphtha and desulfuration [23], [24]. Its main flowchart is demonstrated in Fig. 3.

There are six devices located in the process, which are overhead condenser, reflux accumulator, heat exchanger, head reflux pump, bottom reboiler, and feed pump to the liquified petroleum gas (LPG) splitter. Propane (C3) and butane (C4) are required to be removed in the process. To achieve that, content of butane in the bottom product needs to be aware timely.

Unfortunately, the measurement of butane content is usually obtained by a gas chromatograph, which is located in the overhead of the column. The location of gas chromatograph is far away from the process, and it may cause great time-delay problem. Therefore, it is unpractical to rely on the gas chromatograph for measuring the butane content.

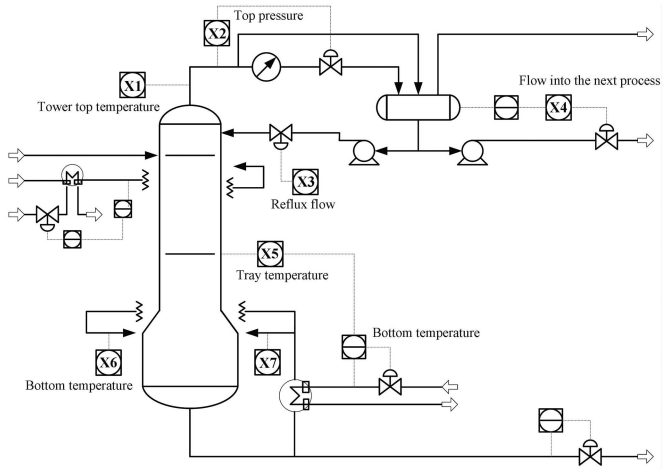

Fig. 3. Basic flowchart of the debutanizer column [23].

TABLE II
Variables Description of Debutanizer Column

Variables	Description
X1	Top temperature
X2	Top pressure
X3	Reflux flow
X4	Flow to next process
X5	6th tray temperature
X6	Bottom temperature A
X7	Bottom temperature B

One feasible way is to construct the mathematical model of butane content, which uses some easy-to-measure variables to predict the hard-to-measure variables, such as butane content. According to the mechanism of debutanizer column, there are seven routinely measured variables have relevant relation with butane content, which are listed in Table II.

This study uses data from the debutanizer column located in Syracuse, Italy. A total of 2394 samples are collected from the process, including the seven easy-to-measure variables and the quality variables. Considering the process dynamics, the final mathematical relation between easy-to-measure variables and quality variables are described as [24]

$$y(k) = f\begin{pmatrix} X(k), X_2(k), \ldots, X_5(k), X_5(k-1), \\ X_5(k-2), X_5(k-3), \frac{X_6(k)+X_7(k)}{2}, \\ y(k-4), y(k-5), y(k-6) \end{pmatrix}. \tag{36}$$

B. Experiments Results

Among all 2394 samples, a quarter is used as the training dataset for establishing the learning models (one third for validation), and the other three thirds are simulated as the testing dataset for application assessment.

For SKPCR, the contribution rate is set 0.9999 and the searching range for kernel parameter is set 1:1:100. Then, the SKPCR model can be established. Table III illustrates the validation error

TABLE III
VALIDATION ERRORS AND PARAMETERS OF SKPCR AT TRAINING STAGE

Layer Number l	Validation Error $e_v^{l)}$	Kernel Parameter c_l
2	0.1517	62
3	0.1373	97
4	0.1371	91
5	0.1373	91

TABLE IV
PREDICTION PERFORMANCES OF DIFFERENT METHODS

	RMSE	R^2
KPCR	0.0253	0.9748
SAE	0.0246	0.9756
Multilayer KPCR	0.0241	0.9770
SSED [14]	0.0227	0.9796
DeKPLS [20]	0.0207	0.9828
SKPCR	0.0185	0.9865

trend and the model parameters at training phase of SKPCR. It can be seen that the validation error continuously declines while the model goes deeper, which means the expression ability is strengthened. When the depth reaches 5, the validation error is larger than that of the previous layer, and the training process is accordingly finished. Otherwise, the deeper model may lead to the overfitting problem. Hence, the three-hidden-layer SKPCR model is established by the training dataset, and it can be directly utilized for the online application stage.

The proposed SKPCR is compared with the state-of-art deep learning networks: SAE and stacked supervised encoder–decoder (SSED) [14]. For those deep networks, the architecture is determined by the trial-and-error method. That is, the architecture obtains the minimum validation error is selected as the eventual architecture (the validation dataset is the same as that used in SKPCR and the searching range of the number of hidden nodes is set 1:1:50). Finally, the structure of deep networks is set [12 8 40 5 1], where 12 and 1 indicates the dimension of input and output variables. The number between 12 and 1 represents the hidden nodes at the corresponding layer. The activation function of deep networks is Sigmoid function, the learning rate is set 1 and the training epochs is set 1000. SAE produces the slightly better performance than that of KPCR, for the reason that deep networks have stronger expression ability due to the complex model structures. However, the pretrain process of SAE is unsupervised, which tries to reconstruct the input data for initializing model parameters. For industrial modeling field, it is the quality variables matter for data modeling, and the mechanism of SAE may not be appropriate. The novel SSED uses quality variables for guiding the pretraining stage, which is verified to be an effective way for quality prediction [14].

Our proposed model is also compared with the novel DeKPLS [20]. DeKPLS is the deep learning version of KPLS, which also belongs to a nonconnectionist deep learning model. Similar to SKPCR, the absence of backpropagation process makes DeKPLS an efficient method in terms of industrial data modeling. The training process of DeKPLS is more complicated than that of SKPCR. Because there are two parameters needed to be configured at each layer, and all parameters are determined by the grid search. Not to mention the process of KPLS iteration for every parameter combination testing. Although the modeling efficiency of DeKPLS has been validated to be better compared with the deep networks, it is expected to be less efficient than that of SKPCR. The searching range of parameters for DeKPLS is set 1:1:50, and the structure of DeKPLS is eventually set a three-hidden-layer (the details of modeling process can be found in [20]).

Table IV lists prediction performances of different methods. KPCR, as a traditional multivariate regression method, gives the worst prediction accuracy with an RMSE of 0.0253. It is obvious that traditional shallow learning machines may not produce the satisfactory modeling performance due to their simple model structure. Although kernel techniques help PCR transform into the nonlinear modeling method, it is difficult to ensure the complex functional relations between variables can be expressed thoroughly by one specific kernel mapping. The state-of-art deep learning networks, SAE and SSED, demonstrate better modeling precision than that of KPCR. Especially for SSED, by utilizing quality variables for guiding the whole pretraining process, its parameter setting is quality oriented. Accordingly, SSED provides a better prediction performance than that of SAE, with an RMSE of 0.0227.

Deep networks have shown their strong learning ability for data modeling. However, the training process is time-consuming and complex. For industrial data modeling, sometimes it is the modeling efficiency play a more important role. The novel DeKPLS performs better in terms of modeling accuracy and efficiency, with the minimum RMSE compared with the deep networks. However, DeKPLS has two parameters at each layer, and the objective of training process is to find the optimal combination of these two parameters, which is very complicated and inefficient. Furthermore, DeKPLS simply stacks several KPLS modules for constructing the deep model. Since KPLS also has the dimension-reduction procedures, the deep features of DeKPLS may have the information loss problem as multilayer KPCR does. Similar to DeKPLS, multilayer KPCR stacks several KPCR modules to build a hierarchical learning model. This design may improve the model complexity and learning ability, which can be proven by the better prediction RMSE of multilayer KPCR. However, the accumulation of PCA process makes deep features relatively uncorrelated to the original data. That makes deep features the improper representation for regression building.

SKPCR achieves the best prediction performance among these methods. The strong learning ability of SKPCR mainly attribute to the reconstructed kernel mapping at each layer. Kernel mapping belongs to the nonlinear mapping form. By stacking multiple kernel mappings, very complicated expressions can be learned. Furthermore, the risk of information loss caused by deep model is compensated by the enhanced inputs at each layer. The detailed prediction results can be found in Fig. 4, and the absolute prediction errors of different models are demonstrated in the box plots of Fig. 5. In the box plots, the narrower the error

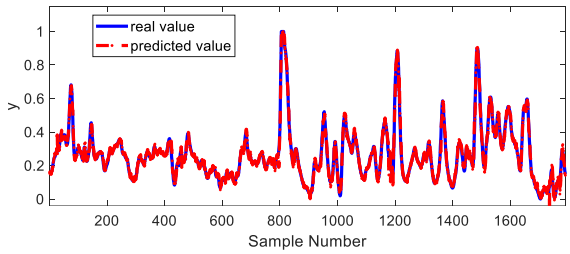

Fig. 4. Detailed prediction results of SKPCR.

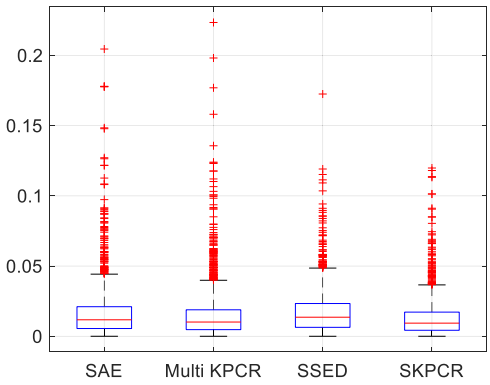

Fig. 5. Box plots of prediction errors of different methods.

TABLE V
COMPUTATIONAL COST OF DIFFERENT METHODS (UNIT IN S)

	CPU Time consumption
SAE	348.5468 s
SSED	348.6093 s
DeKPLS	106.3818 s
SKPCR	28.6599 s

range is, the better the prediction performance is, which means the model's prediction outcomes are closer to the real values.

Modeling efficiency is another significant factor for practical applications. Table V illustrates the computational costs of different deep models. For SAE and SSED, the CPU time consumption is great due to the backpropagation learning mechanism. The model parameters of deep networks are usually with a great amount, and they are fine-tuned repetitively by the backpropagation process. That is the main cost for training a deep network, and the modeling performance is deeply influenced by the initial setting of parameters. The novel DeKPLS abandons the network-based architecture for mining the deep features of data, which makes it much more timesaving than that of deep networks in terms of modeling efficiency. However, the modeling process of DeKPLS is still not concise enough in our opinion. Because the number of parameters at each layer is 2, and the essence of training a DeKPLS is to search the optimal combination of these two parameters. Not to mention that, for each potential combination, the KPLS iteration is executed for evaluating the model performance. For SKPCR, after setting the contribution rate, there is only one parameter needed to be determined for each layer. For this parameter, it can be exhaustively searched, which is not time-consuming due to the noniterative modeling procedures of KPCA. The low computational cost of SKPCR makes it more suitable and applicable for the real implementation in industrial processes.

V. CONCLUSION

This article proposes a deep nonconnectionist learning model based on KPCR, which is referred to as SKPCR. By stacking multiple KPCR modules, a hierarchical feature extraction model is built. Considering the information loss problem, an enhanced learning strategy is designed by reconstructing the inputs for each layer. A real-world industrial case is used to demonstrate the effectiveness, including modeling accuracy and efficiency, of our proposed method.

For industrial soft/smart sensing implementations, artificial intelligence (AI) algorithms are playing more important roles in recent years. With the powerful predictive ability and strong adaptability, AI algorithms can relieve the workers or equipment from extreme working environment. The proposed method has great chance to improve the soft sensing performance for industrial applications. For the reason that the proposed method is much more time-efficient in terms of modeling cost, which is the key factor for real industrial applications.

Although SKPCR has demonstrated superiorities over the state-of-art deep learning methods, there are some challenges for the applications of our proposed method. One major challenge is the configuration of kernel parameters. The modeling performance of SKPCR greatly relies on the values of kernel parameters at each layer. In the present work, the determination of kernel parameters is completed by the grid search method, which belongs to the exhaustive searching method and is relatively time-consuming. This obvious shortcoming restricts the implementation of our proposed method in the industrial field.

In the future, our work will focus on the optimization of model parameter configuration. One feasible way is to utilize the randomized methods for training the kernel learning machine. One of the biggest advantages of randomized methods is that the stochastic configuration of model parameters does not require prior knowledge of the process. Simultaneously, in this way, the exhaustive searching of the potential candidates can be avoided, and the training cost of the model is expected to be reduced to a lower level.

REFERENCES

[1] X. Jiang, X. Kong, and Z. Ge, "Augmented industrial data-driven modeling under the curse of dimensionality," *IEEE/CAA J. Automatica Sinica*, vol. 10, no. 6, pp. 1445–1461, Jun. 2023.
[2] D. Wang and M. J. Felicetti, "Stochastic configuration machines for industrial artificial intelligence," 2023, *arXiv:2308.13570*.
[3] R. Guo, H. Liu, G. Xie, Y. Zhang, and D. Liu, "A self-interpretable soft sensor based on deep learning and multiple attention mechanism: From data selection to sensor modeling," *IEEE Trans. Ind. Informat.*, vol. 19, no. 5, pp. 6859–6871, May 2023.
[4] S. Wold, M. Sjöström, and L. Eriksson, "PLS-regression: A basic tool of chemometrics," *Chemometrics Intell. Lab. Syst.*, vol. 58, no. 2, pp. 109–130, Oct. 2001.
[5] I. Dagher and R. Nachar, "Face recognition using IPCA-ICA algorithm," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 6, pp. 996–1000, Jun. 2006.
[6] Y. Si, Y. Wang, and D. Zhou, "Key-performance-indicator-related process monitoring based on improved kernel partial least squares," *IEEE Trans. Ind. Electron.*, vol. 68, no. 3, pp. 2626–2636, Mar. 2021.

[7] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, vol. 10, no. 5, pp. 1299–1319, Jul. 1998.
[8] R. Rosipal and L. J. Trejo, "Kernel partial least squares regression in reproducing kernel Hilbert space," *J. Mach. Learn. Res.*, vol. 2, pp. 97–123, Dec. 2001.
[9] D. Wang and M. Li, "Stochastic configuration networks: Fundamentals and algorithms," *IEEE Trans. Cybern.*, vol. 47, no. 10, pp. 3466–3479, Oct. 2017.
[10] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.
[11] H. Shao, M. Xia, J. Wan, and C. W. De Silva, "Modified stacked autoencoder using adaptive Morlet wavelet for intelligent fault diagnosis of rotating machinery," *IEEE/ASME Trans. Mechatron.*, vol. 27, no. 1, pp. 24–33, Feb. 2022.
[12] J. Fei and H. Wang, "Experimental investigation of recurrent neural network fractional-order sliding mode control of active power filter," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 67, no. 11, pp. 2522–2526, Nov. 2020.
[13] C. Szegedy et al., "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1–9.
[14] X. Yuan, Y. Gu, Y. Wang, C. Yang, and W. Gui, "A deep supervised learning framework for data-driven soft sensor modeling of industrial processes," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 11, pp. 4737–4746, Nov. 2020.
[15] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao, "Deep learning–based text classification: A comprehensive review," *ACM Comput. Surv.*, vol. 54, no. 3, pp. 1–40, Apr. 2022.
[16] M. J. Felicetti and D. Wang, "Stochastic configuration machines: FPGA implementation," 2023, *arXiv:2310.19225*.
[17] X. Yuan, Y. Gu, and Y. Wang, "Supervised deep belief network for quality prediction in industrial processes," *IEEE Trans. Instrum. Meas.*, vol. 70, Nov. 2020, Art. no. 2503711.
[18] B. Mitchell and J. Sheppard, "Deep structure learning: Beyond connectionist approaches," in *Proc. 11th Int. Conf. Mach. Learn. Appl.*, 2012, pp. 162–167.
[19] Y. Chen and D. Wang, "An improved deep kernel partial least squares and its application to industrial data modeling," *IEEE Trans. Ind. Informat.*, vol. 20, no. 5, pp. 7894–7903, May 2024, doi: 10.1109/TII.2024.3359453.
[20] Y. Chen and X. Deng, "A deep supervised learning framework based on kernel partial least squares for industrial soft sensing," *IEEE Trans. Ind. Informat.*, vol. 19, no. 3, pp. 3178–3187, Mar. 2023.
[21] X. Kong and Z. Ge, "Deep learning of latent variable models for industrial process monitoring," *IEEE Trans. Ind. Informat.*, vol. 18, no. 10, pp. 6778–6788, Oct. 2022.
[22] C. Dachapak, S. Kanae, Z.-J. Yang, and K. Wada, "Kernel principal component regression in reproducing kernel Hilbert space," *Stochastic Syst. Theory Appl.*, vol. 2003, pp. 213–218, May 2003.
[23] L. Fortuna, S. Graziani, and M. G. Xibilia, "Soft sensors for product quality monitoring in debutanizer distillation columns," *Control Eng. Pract.*, vol. 13, no. 4, pp. 499–508, Apr. 2005.
[24] L. Fortuna, S. Graziani, A. Rizzo, and M. G. Xibilia, *Soft Sensors for Monitoring and Control of Industrial Processes.* Berlin, Germany: Springer, 2007.

Yongxuan Chen received the bachelor's degree in measurement and control technology and the master's degree in control engineering from the China University of Petroleum, Qingdao, China, in 2017 and 2020, respectively. He is currently working toward the Ph.D. degree in industrial artificial intelligence with the China University of Mining and Technology, Xuzhou, China.

From 2022 to 2023, he worked as an Assistant with Putian University, Putian, China. His research interests include machine learning techniques for industrial artificial intelligence, stochastic configuration networks, and its applications in industrial data modeling.

Dianhui Wang (Senior Member, IEEE) received the Ph.D. degree in industrial automation from Northeastern University, Shenyang, China, in 1995.

From 1995 to 2001, he worked as a Postdoctoral Fellow with Nanyang Technological University, Singapore, and a Researcher with The Hong Kong Polytechnic University, Hong Kong. He joined La Trobe University, Melbourne, VIC, Australia, in 2001, and worked as a Reader and Associate Professor with the Department of Computer Science and Information Technology until the end of 2020. Since 2017, he has been a Visiting Professor with The State Key Laboratory of Synthetical Automation of Process Industries, Northeastern University. In 2021, he joined the AI Research Institute at China University of Mining and Technology, China, where since December 2021, he has been the Dean and the Founding Director of the Research Center for Stochastic Configuration Machines. His research interests include deep stochastic configuration networks for Big Data analytics in process industries, intelligent sensing system design, and prediction of extreme events.

Dr. Wang is the Editor-in-Chief for *Industrial Artificial Intelligence*, and an Associate Editor for IEEE TRANSACTIONS ON CYBERNETICS, IEEE TRANSACTIONS ON FUZZY SYSTEMS, *Information Sciences, Artificial Intelligence Review, and WIREs Data Ming and Knowledge Discovery*.