Deep Principal Component Analysis Based on Layerwise Feature Extraction and Its Application to Nonlinear Process Monitoring

Xiaogang Deng, Xuemin Tian, Sheng Chen, *Fellow, IEEE*, and Chris J. Harris

***Abstract*— In order to deeply exploit intrinsic data feature information hidden among the process data, an improved kernel principal component analysis (KPCA) method is proposed, which is referred to as deep principal component analysis (DePCA). Specifically, motivated by the deep learning strategy, we design a hierarchical statistical model structure to extract multilayer data features, including both the linear and nonlinear principal components. To reduce the computation complexity in nonlinear feature extraction, the feature-samples' selection technique is applied to build the sparse kernel model for DePCA. To integrate the monitoring statistics at each feature layer, Bayesian inference is used to transform the monitoring statistics into fault probabilities, and then, two probability-based DePCA monitoring statistics are constructed by weighting the fault probabilities at all the feature layers. Two case studies involving a simulated nonlinear system and the benchmark Tennessee Eastman process demonstrate the superior fault detection performance of the proposed DePCA method over the traditional KPCA-based methods.**

***Index Terms*— Bayesian inference, deep learning, kernel principal component analysis (KPCA), nonlinear process monitoring.**

I. Introduction

PROCESS monitoring technologies are playing an increasing role in enhancing industrial plant safety, reducing the production cost and improving the product quality. Since a large amount of process data have been collected and stored in industrial databases, data-based monitoring techniques have attracted a lot of attention in the past two decades [1]–[5]. Typical methods include principal component analysis (PCA) [6], [7], independent component analysis [8]–[10], canonical variate analysis [11], [12], Fisher discriminant analysis [13], [14], and partial least squares [15], [16]. Among these data-based methods, PCA is popular and it has been applied to various industrial processes. By considering the correlation property of time-series process data, Ku *et al.* [17] proposed a dynamic PCA (DPCA) method by augmenting the monitored data vector, while Li *et al.* [18] presented a structured DPCA method to extract correlated latent variables. To reduce the false alarm rate caused by slow normal process changes, Portnoy *et al.* [19] built a recursive PCA method that updates the model parameters using online process data. For monitoring the processes with multiple operation conditions, multimode PCA methods were developed [20]–[22]. A multilevel PCA method was proposed by Liu *et al.* [23] to utilize the prior knowledge on the locations of the process variables. Aiming at multiphase batch process monitoring problems, Zhao and Gao [24] studied an improved PCA with two-step subspace decomposition to explore the between-phase relationship.

All the above-mentioned PCA methods are designed for linear process monitoring. However, most industrial processes are nonlinear, and therefore, researchers have developed nonlinear PCA algorithms, known as kernel PCA (KPCA) [25], [26]. Because of its effectiveness, KPCA has become a state of the art in nonlinear process monitoring. Lee *et al.* [26] first constructed the KPCA-based fault detection method, and Choi *et al.* [27] developed a fault identification index for KPCA-based process monitoring. To analyze the multiscale property of process data, Zhang and Ma [28] proposed a multiscale KPCA method by utilizing wavelet transformation, while Yi *et al.* [29] developed a transformation matrix-based KPCA method to extract the representative and meaningful features from data. Deng *et al.* [30] developed a modified KPCA method that integrates local data structure analysis with KPCA. Jiang and Yan [31] proposed a multiblock KPCA method for nonlinear plant-wide processes by applying mutual information-based clustering to divide the measured variables into multiple subblocks. To enhance the detectability of small

Manuscript received March 13, 2018; revised July 5, 2018; accepted August 8, 2018. Date of publication September 6, 2018; date of current version October 9, 2019. Manuscript received in final form August 9, 2018. This work was supported in part by the Natural Science Foundation of Shandong Province, China, under Grant ZR2014FL016 and Grant ZR2016FQ21, in part by the National Natural Science Foundation of China under Grant 61403418, Grant 21606256, and Grant 61273160, in part by the Fundamental Research Funds for the Central Universities under Grant 17CX02054, and in part by the Shandong Provincial Research and Development Programme under Grant 2018GGX101025. This paper was presented in part at the 2017 International Joint Conference on Neural Networks. Recommended by Associate Editor P. Mhaskar. *(Corresponding author: Xiaogang Deng.)*

X. Deng and X. Tian are with the College of Information and Control Engineering, China University of Petroleum, Dongying 266580, China (e-mail: dengxg2002@gmail.com; tianxm@upc.edu.cn).

S. Chen is with the School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, U.K., and also with King Abdulaziz University, Jeddah 21589, Saudi Arabia (e-mail: sqc@ecs.soton.ac.uk).

C. J. Harris is with the School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, U.K. (e-mail: cjh@ecs.soton.ac.uk).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCST.2018.2865413

process disturbance, Cheng *et al.* [32] utilized multivariate exponentially moving average to estimate the process mean shifts, and integrated the predicted shift with the KPCA method. More works on KPCA-based process monitoring can be found in [33]–[35]. A recent work [36] proposed a novel serial PCA (SPCA) method by closely integrating PCA and KPCA based on a serial model structure.

All the KPCA methods mentioned above use KPCA as the core algorithm, and they have been applied to various nonlinear process monitoring cases. However, there are some open problems worthy of further investigating. One important problem is: is the KPCA model sufficient to extract the intrinsic features of process data? KPCA applies the kernel function to implement nonlinear transformation and obtains only one layer of nonlinear features for statistical modeling. However, it is often difficult to mine the total information by the single feature extraction step for the complicated nonlinear process data. In other words, the feature extraction of KPCA may be "shallow" and unable to exploit the intrinsic data information sufficiently. Even the novel SPCA [36], which consists of a layer of linear PCA feature extraction followed by a layer of nonlinear KPCA feature extraction, may still be insufficiently "deep." In recent years, deep learning has achieved great success in many fields of science and engineering [37]–[44]. Deep learning theory indicates that the multilevel feature extraction is beneficial to discover the intricate data structure. Therefore, to build a multilayer feature extraction model is of great value to improve the KPCA-based process monitoring method.

Motivated by the above analysis, we proposes an improved KPCA method with a deep architecture, referred to as deep PCA (DePCA), which adopts the layerwise feature extraction strategy to obtain the multilevel data features for nonlinear process monitoring. Different from the traditional PCA and KPCA with only one layer of features as well as unlike the SPCA [36] that utilizes only one layer of linear features and one layer of nonlinear features, DePCA constructs a deep feature extraction model to capture multiple layers of linear and nonlinear features hierarchically. As DePCA uses PCA and KPCA as feature extraction tools, complex nonlinear network optimization is avoided in the procedure of multilayer feature extraction. Because multiple kernel modeling increases the computational complexity, the feature-samples' selection (FSS) technique [45] is applied in the proposed DePCA method to build the sparse kernel model. At each layer, two monitoring statistics are constructed based on the linear or nonlinear features. Furthermore, Bayesian inference integrates all the monitoring statistics from different layers to yield an overall indication on the process status. To the best of our knowledge, there is no study to date combining the deep learning technology and the KPCA method together in the process monitoring and fault diagnosis field, and we are the first to propose the deep learning-enhanced KPCA method for nonlinear process monitoring.

The remainder of this paper is structured as follows. In Section II, a brief overview of PCA, KPCA, and SPCA is given, while our proposed DePCA strategy for nonlinear process monitoring is detailed in Section III. In Section IV, two case studies are used to validate the proposed method, and our conclusions are drawn in Section V.

II. Overview of PCA, KPCA, and SPCA

A. PCA Method

Given a training data matrix $\boldsymbol{X} \in \mathbb{R}^{N\times M}$ with N samples of M variables, which is assumed to be mean-centered and variance-scaled, PCA seeks a projection vector $\boldsymbol{p} \in \mathbb{R}^M$ such that the linear transformation $\boldsymbol{t} = \boldsymbol{X}\boldsymbol{p}$ has the maximal variance. This problem can be formulated as

$$\max_{\boldsymbol{p}} \frac{1}{N-1}\boldsymbol{t}^T\boldsymbol{t} = \max_{\boldsymbol{p}} \frac{1}{N-1}\boldsymbol{p}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{p} \quad \text{s.t. } \boldsymbol{p}^T\boldsymbol{p} = 1 \tag{1}$$

where the principal component (PC) vector $\boldsymbol{t} \in \mathbb{R}^N$ is also called the score vector. Solving the optimization task (1) leads to the eigenvalue decomposition as

$$\lambda\boldsymbol{p} = \frac{1}{N-1}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{p}. \tag{2}$$

The solution to (2) brings the M eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_M$ with the corresponding eigenvectors $\boldsymbol{p}_i$ for $1 \leq i \leq M$. Each $\boldsymbol{p}_i$ is also called a loading vector. All M projection directions $\boldsymbol{P} = [\boldsymbol{p}_1\ \boldsymbol{p}_2 \ldots \boldsymbol{p}_M] \in \mathbb{R}^{M\times M}$ can be divided into two groups: the first K directions $\boldsymbol{P}_P = [\boldsymbol{p}_1\ \boldsymbol{p}_2 \ldots \boldsymbol{p}_K] \in \mathbb{R}^{M\times K}$ represent the principal component subspace (PCS), while the other $M-K$ directions $\boldsymbol{P}_R = [\boldsymbol{p}_{K+1}\ \boldsymbol{p}_{K+2} \ldots \boldsymbol{p}_M] \in \mathbb{R}^{M\times(M-K)}$ construct the residual subspace (RS). Fig. 1(a) depicts the PCA model training, which captures the linear features $\boldsymbol{T} = \boldsymbol{X}\boldsymbol{P}$ for $\boldsymbol{X}$ by optimizing $\boldsymbol{P}$.

For the testing vector $\boldsymbol{x}_t = [x_{t,1}\ x_{t,2} \ldots x_{t,M}]^T \in \mathbb{R}^M$ at the time instant t, its ith score $t_{t,i}$ is computed as

$$t_{t,i} = \boldsymbol{x}_t^T\boldsymbol{p}_i, \quad 1 \leq i \leq M. \tag{3}$$

Given $\boldsymbol{t}_{tK} = [t_{t,1}\ t_{t,2} \ldots t_{t,K}]^T \in \mathbb{R}^K$, the following T^2 and Q monitoring statistics can be built for fault detection [46]:

$$T^2 = \boldsymbol{t}_{tK}^T\boldsymbol{\Lambda}^{-1}\boldsymbol{t}_{tK} \tag{4}$$

$$Q = \|\boldsymbol{x}_t - \widehat{\boldsymbol{x}}_t\|^2 = (\boldsymbol{x}_t - \boldsymbol{P}_P\boldsymbol{t}_{tK})^T(\boldsymbol{x}_t - \boldsymbol{P}_P\boldsymbol{t}_{tK}) \tag{5}$$

where $\boldsymbol{\Lambda}$ is the $K \times K$ diagonal matrix with the eigenvalues λ_i for $1 \leq i \leq K$ as its diagonal elements and $\widehat{\boldsymbol{x}}_t = \boldsymbol{P}_P\boldsymbol{t}_{tK}$ is the reconstruction of the testing vector $\boldsymbol{x}_t$. It can be seen that for the PCA method, the T^2 and Q statistics are constructed based on the linear features of the data. Specifically, the T^2 statistic is designed to monitor the data variations in the PCS, while the Q statistic is used to monitor the data changes in the RS. With these two statistics, therefore, the data changes, more specifically, the changes in the linear features, can be monitored comprehensively. The PCA-based online monitoring is shown in Fig. 1(b), which involves the original data layer L1 and the linear feature layer L2. The PC number K is a key parameter in PCA, and many methods have been discussed to determine its value [17], [26]. In this paper, the average eigenvalue method is adopted, which retains the PCs whose eigenvalues are larger than the average eigenvalue.

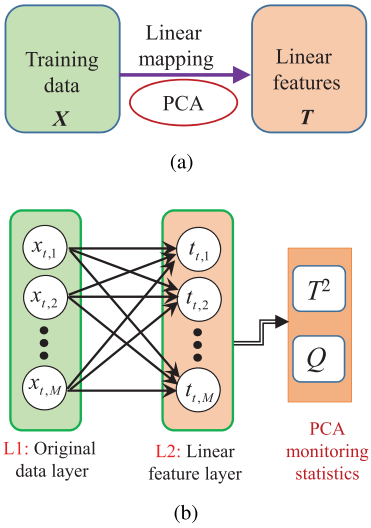


Fig. 1. Schematic for PCA-based process monitoring. (a) PCA model training. (b) PCA online monitoring.

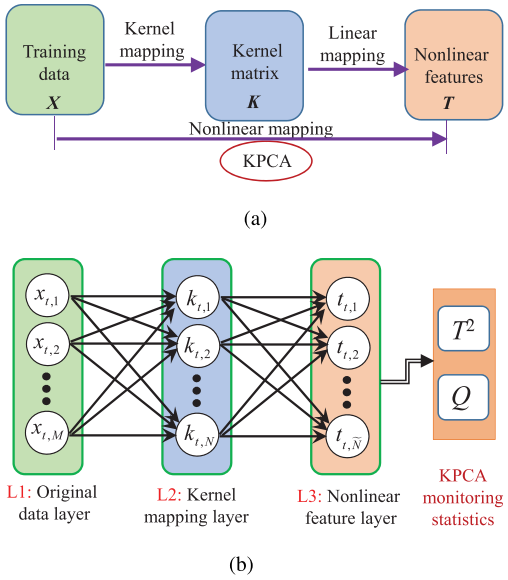


Fig. 2. Schematic for KPCA-based process monitoring. (a) KPCA model training. (b) KPCA online monitoring.

B. KPCA Method

A nonlinear mapping $\boldsymbol{\Phi}(.)$ projects the training data $\boldsymbol{X}$ onto a new high-dimensional space $\mathcal{F}$. Then, linear PCA is performed on the new data matrix $\boldsymbol{\Phi}(\boldsymbol{X}) \in \mathbb{R}^N \times \mathcal{F}$ to find the loading vector $\boldsymbol{p} \in \mathcal{F}$, so that the score vector $\boldsymbol{t} = \boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{p}$ has the maximal variance. This leads to the optimization

$$\max_{\boldsymbol{p}} \frac{\boldsymbol{t}^T\boldsymbol{t}}{N-1} = \max_{\boldsymbol{p}} \frac{\boldsymbol{p}^T\boldsymbol{\Phi}^T(\boldsymbol{X})\boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{p}}{N-1}$$
$$\text{s.t.} \quad \boldsymbol{p}^T\boldsymbol{p} = 1. \tag{6}$$

Denoting $\boldsymbol{X} = [\boldsymbol{x}_1 \ \boldsymbol{x}_2 \ldots \boldsymbol{x}_N]^{\mathrm{T}}$ and then according to [25] and [26], there exists a coefficient vector $\boldsymbol{\alpha} = [\alpha_1 \ \alpha_2 \ldots \alpha_N]^T \in \mathbb{R}^N$ satisfying

$$\boldsymbol{p} = \sum_{j=1}^{N} \boldsymbol{\Phi}(\boldsymbol{x}_j)\alpha_j = \boldsymbol{\Phi}^T(\boldsymbol{X})\boldsymbol{\alpha}. \tag{7}$$

By combining (6) and (7), the KPCA optimization becomes

$$\max_{\boldsymbol{\alpha}} \frac{1}{N-1}\boldsymbol{\alpha}^T\boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{\Phi}^T(\boldsymbol{X})\boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{\Phi}^T(\boldsymbol{X})\boldsymbol{\alpha}$$
$$\text{s.t.} \ \boldsymbol{\alpha}^T\boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{\Phi}^T(\boldsymbol{X})\boldsymbol{\alpha} = 1. \tag{8}$$

To avoid defining nonlinear mapping $\Phi(.)$, a kernel matrix is introduced as $\boldsymbol{K} = \boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{\Phi}^T(\boldsymbol{X})$. With this kernel trick [25], the (i, j)th element of $\boldsymbol{K}$ is computed according to

$$\boldsymbol{K}(i, j) = \boldsymbol{\Phi}^T(\boldsymbol{x}_i)\boldsymbol{\Phi}(\boldsymbol{x}_j) = \ker(\boldsymbol{x}_i, \boldsymbol{x}_j). \tag{9}$$

Typical kernel functions are the Gaussian kernel $\ker(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/\sigma)$, where $\sigma > 0$ is the kernel width, and the polynomial kernel $\ker(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i^T\boldsymbol{x}_j + d_0)^{d_1}$, where d_0 and d_1 are the polynomial kernel parameters [25], [26], [47]. Thus, solving (8) leads to the eigenvalue decomposition as

$$\boldsymbol{K}\boldsymbol{\alpha} = (N-1)\lambda\boldsymbol{\alpha}. \tag{10}$$

The solutions of (10) bring the $\tilde{N}$ nonzero eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \lambda_{\tilde{N}}$ with the corresponding eigenvectors $\boldsymbol{\alpha}_i$ for $1 \leq i \leq \tilde{N}$, where $\tilde{N} \leq N$. Similar to PCA, all the eigenvectors $\boldsymbol{A} = [\boldsymbol{\alpha}_1 \ \boldsymbol{\alpha}_2 \ldots \boldsymbol{\alpha}_{\tilde{N}}] \in \mathbb{R}^{N\times\tilde{N}}$ are divided into two parts, where the first K eigenvectors, whose corresponding eigenvalues are no less than the average eigenvalue, define the kernel PCS, while the other $\tilde{N} - K$ eigenvectors construct the kernel RS. As shown in Fig. 2(a), KPCA model training computes the kernel PCs (KPCs) $\boldsymbol{T} = \boldsymbol{K}\boldsymbol{A} \in \mathbb{R}^{N\times\tilde{N}}$ as the nonlinear features of $\boldsymbol{X}$.

The ith kernel score for the testing vector $\boldsymbol{x}_t \in \mathbb{R}^M$ is computed as

$$t_{t,i} = \boldsymbol{\Phi}^T(\boldsymbol{x}_t)\boldsymbol{p}_i = \boldsymbol{\Phi}^T(\boldsymbol{x}_t)\boldsymbol{\Phi}^T(\boldsymbol{X})\boldsymbol{\alpha}_i = \boldsymbol{k}_t^T\boldsymbol{\alpha}_i \tag{11}$$

where $\boldsymbol{k}_t = [k_{t,1} \ k_{t,2} \ldots k_{t,N}]^T = \boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{\Phi}(\boldsymbol{x}_t) \in \mathbb{R}^N$ and $k_{t,j} = \ker(\boldsymbol{x}_j, \boldsymbol{x}_t)$ for $1 \leq j \leq N$. To monitor the change in nonlinear features, two monitoring statistics are used [26]

$$T^2 = \boldsymbol{t}_{tK}^T\boldsymbol{\Lambda}^{-1}\boldsymbol{t}_{tK} \tag{12}$$

$$Q = \|\boldsymbol{\Phi}(\boldsymbol{x}_t) - \widehat{\boldsymbol{\Phi}}(\boldsymbol{x}_t)\|^2 = \sum_{i=1}^{\tilde{N}} t_{t,i}^2 - \sum_{i=1}^{K} t_{t,i}^2 \tag{13}$$

where $\boldsymbol{t}_{tK} = [t_{t,1} \ t_{t,2} \ldots t_{t,K}]^T$, $\boldsymbol{\Lambda}$ is the $K \times K$ diagonal matrix with the eigenvalues λ_i for $1 \leq i \leq K$ as its diagonal elements, and $\widehat{\boldsymbol{\Phi}}(\boldsymbol{x}_t)$ is the reconstruction of the vector $\boldsymbol{\Phi}(\boldsymbol{x}_t)$. The online monitoring of $\boldsymbol{x}_t$ is depicted in Fig. 2(b), which has a three-layer structure involving the original data layer L1, the kernel mapping layer L2, and the nonlinear feature layer L3. Similar to PCA, KPCA only monitors one layer of features. However, the difference is that KPCA introduces a kernel mapping layer to obtain nonlinear features.

C. SPCA Method

The SPCA [36] integrates PCA and KPCA by a serial model structure to extract both linear and nonlinear features for process monitoring. Specifically, the SPCA model training

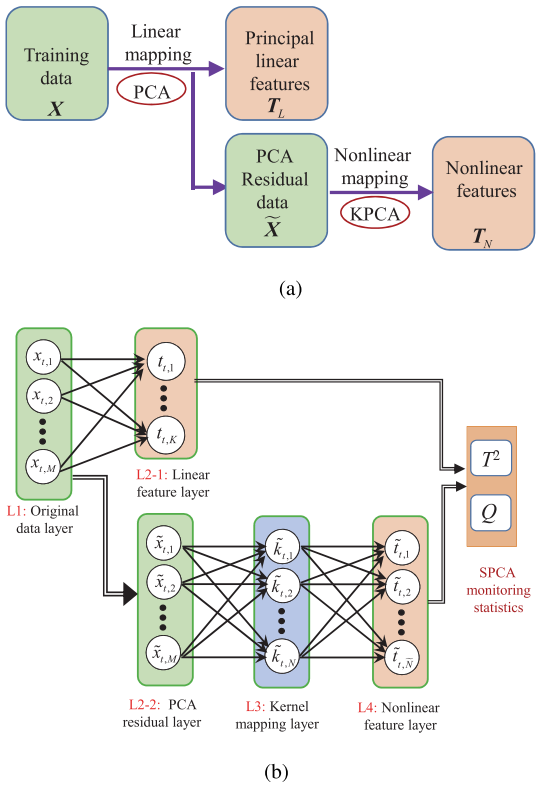


Fig. 3. Schematic for SPCA-based process monitoring. (a) SPCA model training. (b) SPCA online monitoring.

is shown in Fig. 3(a). Given the training data $\boldsymbol{X}$, the linear PCA decomposition is first performed as

$$\boldsymbol{X} = \sum_{i=1}^{K} \boldsymbol{t}_i \boldsymbol{p}_i^T + \widetilde{\boldsymbol{X}} \tag{14}$$

where $\boldsymbol{t}_i$ is the ith linear score vector, $\boldsymbol{p}_i$ is the corresponding loading vector, K is the number of PCs retained, and $\widetilde{\boldsymbol{X}} = [\widetilde{\boldsymbol{x}}_1\ \widetilde{\boldsymbol{x}}_2 \ldots \widetilde{\boldsymbol{x}}_N]^{\mathrm{T}}$ is the PCA residual data matrix. Then, the KPCA is applied to $\widetilde{\boldsymbol{X}}$ to obtain a KPCA decomposition as

$$\boldsymbol{\Phi}\big(\widetilde{\boldsymbol{X}}\big) = \sum_{i=1}^{\widetilde{K}} \widetilde{\boldsymbol{t}}_i \widetilde{\boldsymbol{p}}_i^T + \boldsymbol{E} \tag{15}$$

where $\widetilde{\boldsymbol{t}}_i \in \mathbb{R}^N$ is the ith nonlinear score vector or feature, $\widetilde{\boldsymbol{p}}_i \in \mathcal{F}$ is the corresponding loading vector in KPCA decomposition, and $\widetilde{K}$ is the number of KPCs retained, while $\boldsymbol{E} \in \mathbb{R}^N \times \mathcal{F}$ is the KPCA residual matrix. Thus, SPCA first utilizes the PCA transformation to calculate the PCs $\boldsymbol{T}_L = [\boldsymbol{t}_1\ \boldsymbol{t}_2 \ldots \boldsymbol{t}_K]$ as the linear features. Then, KPCA modeling is implemented on the residual matrix $\widetilde{\boldsymbol{X}}$ to compute the KPCs $\boldsymbol{T}_N = [\widetilde{\boldsymbol{t}}_1\ \widetilde{\boldsymbol{t}}_2 \ldots \widetilde{\boldsymbol{t}}_{\widetilde{N}}]$ as the nonlinear features. The determination of $\boldsymbol{T}_N$ is based on the same kernel trick with the corresponding eigenvectors $\widetilde{\boldsymbol{\alpha}}_i \in \mathbb{R}^N$ for $1 \le i \le \widetilde{N}$.

The SPCA online monitoring for the testing sample $\boldsymbol{x}_t$ is depicted in Fig. 3(b), which has a four-layer structure involving the original data layer L1, the linear feature and PCA residual layer L2, the kernel mapping layer L3, and the nonlinear feature layer L4. More specifically, for the testing vector $\boldsymbol{x}_t$, its ith linear score is given by

$$t_{t,i} = \boldsymbol{x}_t^T \boldsymbol{p}_i. \tag{16}$$

The first K scores $[t_{t,1}\ t_{t,2} \ldots t_{t,K}]^T$ are used as the linear features of $\boldsymbol{x}_t$, and the residual vector of $\boldsymbol{x}_t$ is given by

$$\widetilde{\boldsymbol{x}}_t = \boldsymbol{x}_t - \sum_{i=1}^{K} t_{t,i} \boldsymbol{p}_i. \tag{17}$$

The ith nonlinear score of $\widetilde{\boldsymbol{x}}_t$ is extracted as

$$\widetilde{t}_{t,i} = \boldsymbol{\Phi}^T(\widetilde{\boldsymbol{x}}_t)\widetilde{\boldsymbol{p}}_i = \widetilde{\boldsymbol{k}}_t^T \widetilde{\boldsymbol{\alpha}}_i, \quad 1 \le i \le \widetilde{N} \tag{18}$$

where $\widetilde{\boldsymbol{k}}_t \in \mathbb{R}^N$ is the test kernel vector whose jth element is $\ker(\widetilde{\boldsymbol{x}}_j, \widetilde{\boldsymbol{x}}_t)$. The first $\widetilde{K}$ nonlinear scores together with the first K linear scores form the combined linear and nonlinear feature vector $\boldsymbol{t}_{\mathrm{SPCA}} = [t_{t,1}\ t_{t,2} \ldots t_{t,K}\ \widetilde{t}_{t,1}\ \widetilde{t}_{t,2} \ldots \widetilde{t}_{t,\widetilde{K}}]^T$ to construct the two monitoring statistics as

$$T^2 = \boldsymbol{t}_{\mathrm{SPCA}}^T \boldsymbol{\Gamma}^{-1} \boldsymbol{t}_{\mathrm{SPCA}} \tag{19}$$

$$Q = \sum_{i=1}^{\widetilde{N}} (\widetilde{t}_{t,i})^2 - \sum_{i=1}^{\widetilde{K}} (\widetilde{t}_{t,i})^2 \tag{20}$$

where $\boldsymbol{\Gamma}$ is the covariance matrix of the feature vector $\boldsymbol{t}_{\mathrm{SPCA}}$ computed under the normal operating condition.

III. Proposed DePCA Method

KPCA performs deeper data learning than PCA and it outperforms PCA in many application cases [26], [27]. SPCA constructs a deeper monitoring model than KPCA, and it has been shown to offer better monitoring performance than KPCA [36]. It can be seen that a deeper data learning structure is beneficial to improve nonlinear process monitoring performance. Deep learning theory [37]–[44] has been applied to construct deep neural networks, including convolutional deep neural networks [42], deep belief networks [43], and autoencoder deep networks [44]. These deep neural networks consist of multiple nonlinear feature extraction layers and transforms the lower level features into higher level features layerwise. It turns out that deep learning can discover intricate structures in high-dimensional data, while conventional shallow learning may miss. With many simple but nonlinear feature layers, very complex nonlinear feature extraction is achieved. Motivated by the success of deep learning as well as inspired by the above layered structure analysis of PCA, KPCA, and SPCA, we propose a deep PCA method with layerwise feature extraction to improve fault detection performance.

A. Construction of DePCA Model

The proposed DePCA modeling extracts linear and nonlinear features by integrating the PCA and KPCA in an L-layered structure as shown in Fig. 4(a). In this deep layerwise feature extraction structure, the linear PCs are first extracted from the training data $\boldsymbol{X} \in \mathbb{R}^{N \times M}$ as the first-layer features $\boldsymbol{T}^{(1)} \in \mathbb{R}^{N \times \widetilde{N}^{(1)}}$, where $\widetilde{N}^{(1)} = M$. Then KPCA is applied sequentially to $\boldsymbol{T}^{(l)}$ to extract the nonlinear KPCs $\boldsymbol{T}^{(l+1)} \in \mathbb{R}^{N \times \widetilde{N}^{(l+1)}}$ for $1 \le l \le L-1$, where $\widetilde{N}^{(l+1)}$ is the number of nonlinear features extracted at the $(l+1)$th layer.

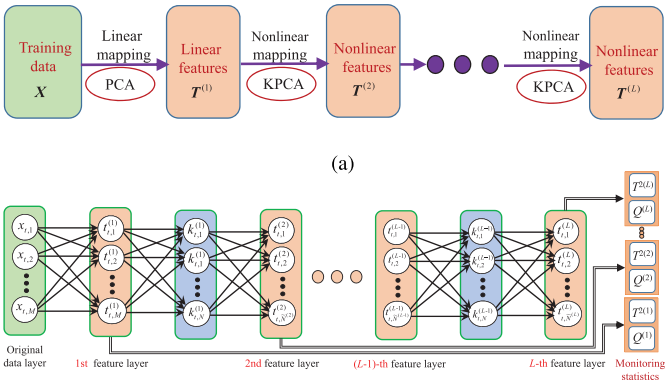


Fig. 4. Schematic for DePCA-based process monitoring. (a) DePCA model training. (b) DePCA online monitoring.

More specifically, at the first feature layer, a linear optimization task is designed as

$$\max_{\boldsymbol{p}^{(1)}} \frac{1}{N-1}(\boldsymbol{p}^{(1)})^T \boldsymbol{X}^T \boldsymbol{X}\boldsymbol{p}^{(1)} \quad \text{s.t. } (\boldsymbol{p}^{(1)})^T \boldsymbol{p}^{(1)} = 1 \tag{21}$$

to produce the M projection vectors $\boldsymbol{p}_i^{(1)} \in \mathbb{R}^M$ and the corresponding score vectors $\boldsymbol{t}_i^{(1)} = \boldsymbol{X}\boldsymbol{p}_i^{(1)}, 1 \le i \le M$, which constitute the first-layer features $\boldsymbol{T}^{(1)} = [\boldsymbol{t}_1^{(1)}\ \boldsymbol{t}_2^{(1)} \dots \boldsymbol{t}_M^{(1)}]$.

At the lth feature layer for $2 \le l \le L$, we execute the nonlinear optimization task

$$\max_{\boldsymbol{p}^{(l)}} \frac{1}{N-1}(\boldsymbol{p}^{(l)})^T \boldsymbol{\Phi}^T(\boldsymbol{T}^{(l-1)})\boldsymbol{\Phi}(\boldsymbol{T}^{(l-1)})\boldsymbol{p}^{(l)} \quad \text{s.t. } (\boldsymbol{p}^{(l)})^T \boldsymbol{p}^{(l)} = 1 \tag{22}$$

in which the $(l-1)$th-layer feature matrix $\boldsymbol{T}^{(l-1)}$ is the training data for the lth layer, and $\boldsymbol{p}^{(l)}$ is the nonlinear projection vector that can be expressed according to (7) as

$$\boldsymbol{p}^{(l)} = \sum_{j=1}^{N} \boldsymbol{\Phi}^T(\overline{\boldsymbol{t}}_j^{(l-1)})\alpha_j^{(l)} = \boldsymbol{\Phi}^T(\boldsymbol{T}^{(l-1)})\boldsymbol{\alpha}^{(l)} \tag{23}$$

where $\overline{\boldsymbol{t}}_j^{(l-1)} \in \mathbb{R}^{\widetilde{N}^{(l-1)}}$ denotes the jth column of $(\boldsymbol{T}^{(l-1)})^T$ and $\boldsymbol{\alpha}^{(l)} = [\alpha_1^{(l)}\ \alpha_2^{(l)} \dots \alpha_N^{(l)}]^T$. Furthermore, denote $\boldsymbol{K}^{(l-1)} = \boldsymbol{\Phi}(\boldsymbol{T}^{(l-1)})\boldsymbol{\Phi}^T(\boldsymbol{T}^{(l-1)})$. The nonlinear optimization (22) can be written in the kernel form

$$\max_{\boldsymbol{\alpha}^{(l)}} \frac{1}{N-1}(\boldsymbol{\alpha}^{(l)})^T \boldsymbol{K}^{(l-1)}\boldsymbol{K}^{(l-1)}\boldsymbol{\alpha}^{(l)} \quad \text{s.t. } (\boldsymbol{\alpha}^{(l)})^T \boldsymbol{K}^{(l-1)}\boldsymbol{\alpha}^{(l)} = 1 \tag{24}$$

which yields the projection vectors $\boldsymbol{\alpha}_i^{(l)} \in \mathbb{R}^N$ and the corresponding nonlinear score vectors $\boldsymbol{t}_i^{(l)} = \boldsymbol{K}^{(l-1)}\boldsymbol{\alpha}_i^{(l)} \in \mathbb{R}^N$ for $1 \le i \le \widetilde{N}^{(l)}$, where $\widetilde{N}^{(l)}$ is the number of $\boldsymbol{\alpha}_i^{(l)}$ corresponding to nonzero eigenvalues. Therefore, the lth layer features are obtained as $\boldsymbol{T}^{(l)} = [\boldsymbol{t}_1^{(l)}\ \boldsymbol{t}_2^{(l)} \dots \boldsymbol{t}_{\widetilde{N}^{(l)}}^{(l)}] \in \mathbb{R}^{N \times \widetilde{N}^{(l)}}$.

The online monitoring procedure for the testing vector $\boldsymbol{x}_t$ using the trained DePCA model is given in Fig. 4(b), where

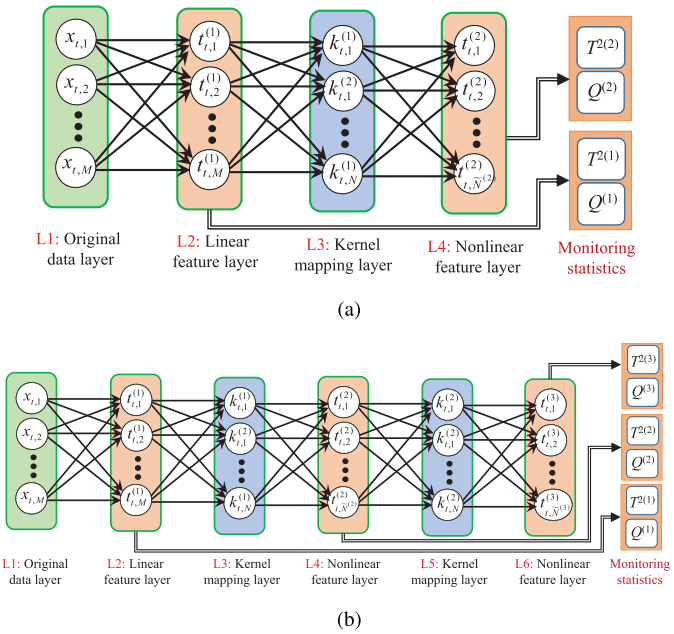


Fig. 5. Schematic of DePCA-N1- and DePCA-N2-based process monitoring. (a) DePCA-N1 online monitoring. (b) DePCA-N2 online monitoring.

$\boldsymbol{t}_t^{(1)} = [t_{t,1}^{(1)}\ t_{t,2}^{(1)} \dots t_{t,M}^{(1)}]^T$ is the first-layer feature vector computed by

$$t_{t,i}^{(1)} = \boldsymbol{x}_t^T \boldsymbol{p}_i^{(1)}, \quad 1 \le i \le M. \tag{25}$$

The $l$thlayer feature vector $\boldsymbol{t}_t^{(l)} = [t_{t,1}^{(l)}\ t_{t,2}^{(l)} \dots t_{t,\widetilde{N}^{(l)}}^{(l)}]^T$, where $2 \le l \le L$, is expressed as

$$t_{t,i}^{(l)} = \left(\boldsymbol{k}_t^{(l-1)}\right)^T \boldsymbol{\alpha}_i^{(l)} \tag{26}$$

in which the kernel vector $\boldsymbol{k}_t^{(l-1)} = \boldsymbol{\Phi}(\boldsymbol{T}^{(l-1)})\boldsymbol{\Phi}(\boldsymbol{t}_t^{(l-1)}) = [k_{t,1}^{(l-1)}\ k_{t,2}^{(l-1)} \dots k_{t,N}^{(l-1)}]^T \in \mathbb{R}^N$. It is worth emphasizing that unlike the multilayer neural network, the DePCA method realizes the layer-by-layer feature extraction without the need for complicated network optimization.

How many layers are appropriate is influenced by many complex factors and is certainly problem-dependent. In the application to process monitoring, we implement two specific DePCA models by setting $L = 2$ and $L = 3$, which are denoted as DePCA-N1 and DePCA-N2, respectively, where N1 indicates that the number of nonlinear feature layers is one, while N2 indicates that the number of nonlinear feature layers is two. For the testing vector $\boldsymbol{x}_t$, the monitoring procedure based on DePCA-N1 is illustrated in Fig. 5(a), which has the four-layer structure with the original data layer L1, the linear feature layer L2, the kernel mapping layer L3, and the nonlinear feature layer L4. Note that DePCA-N1 is very different from SPCA. In DePCA-N1, we adopt the Gaussian kernel function for the kernel mapping layer L3. In some applications, the two layers of linear and nonlinear features in DePCA-N1 may be insufficient to mine the data features, and the deeper DePCA-N2 with six layers may be preferred, which is illustrated in Fig. 5(b). During online monitoring, DePCA-N2 extracts three layers of features, including one layer of linear features and two layers of nonlinear features. To ensure

the diversity of the nonlinear features, the kernel mapping layer L3 adopts the two-order polynomial kernel, while the kernel mapping layer L5 employs the Gaussian kernel.

Remark 1: It is clear that the proposed DePCA method is very different from our previous SPCA method. For simplicity and clarity, let us compare the DePCA-N1, which consists of linear-feature extraction and nonlinear-feature extraction, with the SPCA, which also has linear-feature extraction and nonlinear-feature extraction. In the SPCA, the KPCA is only applied to the reconstructed residual matrix, i.e., only applied to the RS. Therefore, information contained in the PC matrix of the PCA is not exploited. In contrast, in the DePCA-N1, the KPCA is applied to both the PCS and the RS, i.e., the whole original data space. Therefore, there is no information loss in this KPCA. Consequently, the DePCA-N1 generally can provide better feature extraction performance than our early SPCA.

Before applying DePCA-N1 and DePCA-N2 to fault detection, two issues need to be discussed. The first one is computational complexity. DePCA-N1 and DePCA-N2 apply more feature layers than KPCA. Especially, DePCA-N2 involves two KPCA modeling procedures. The dimensions of the kernel vector $\boldsymbol{k}_t^{(l)}$ at the L3 and L5 layers are equal to the number of training samples N. In most cases, N is very large and this may lead to high computational loads at the feature layers. Therefore, it is necessary to build a sparse kernel model for DePCA. The other issue is how to construct online monitoring statistics. Since each feature layer provides two monitoring statistics, how to combine the monitoring results at different layers into an overall monitoring index is also important for ensuring good monitoring performance. These two problems will be discussed in Sections III-B and III-C.

B. Sparse Kernel Model With Feature-Samples' Selection

From various sparse kernel modeling approaches [45], [47]–[53], we adopt the FSS method of [45]. In (24) of DePCA modeling, all the samples $\boldsymbol{\Phi}(\overline{\boldsymbol{t}}_j^{(l-1)})$ for $1 \le j \le N$ are used to construct the projection vector $\boldsymbol{p}^{(l)}$, which results in the high-dimensional $N \times N$ kernel matrix, leading to a high computational cost in kernel modeling and online monitoring, particularly for large N. In fact, it is unnecessary to use all the samples in the construction of eigenvector (23) and a subset $\boldsymbol{T}_S^{(l-1)}$ is often sufficient to approximate the data space as

$$\boldsymbol{p}^{(l)} \approx \boldsymbol{\Phi}^T\left(\boldsymbol{T}_S^{(l-1)}\right)\boldsymbol{\alpha}_S^{(l)} \tag{27}$$

where $\boldsymbol{T}_S^{(l-1)} = [\overline{\boldsymbol{t}}_{S,1}\ \overline{\boldsymbol{t}}_{S,2}\ldots\overline{\boldsymbol{t}}_{S,N_S^{(l-1)}}]^T$ for $N_S^{(l-1)} \le N$ is the subset of $\boldsymbol{T}^{(l-1)}$, called the feature-samples set, and $\boldsymbol{\alpha}_S^{(l)} \in \mathbb{R}^{N_S^{(l-1)}}$ is the coefficient vector based on the feature samples. For convenience of discussion, denote $\boldsymbol{Y} = \boldsymbol{\Phi}(\boldsymbol{T}^{(l-1)}) \in \mathbb{R}^N \times \mathcal{F}$, $\boldsymbol{Y}_S = \boldsymbol{\Phi}(\boldsymbol{T}_S^{(l-1)}) \in \mathbb{R}^{N_S^{(l-1)}} \times \mathcal{F}$, $\boldsymbol{y}_j = \boldsymbol{\Phi}(\overline{\boldsymbol{t}}_j^{(l-1)}) \in \mathcal{F}$, and $\boldsymbol{y}_{S,j} = \boldsymbol{\Phi}(\overline{\boldsymbol{t}}_{S,j}) \in \mathcal{F}$. Then, the estimate of $\boldsymbol{y}_j$ can be obtained by the feature samples as

$$\widehat{\boldsymbol{y}}_j = \boldsymbol{Y}_S^T\boldsymbol{\beta}_j \tag{28}$$

where $\boldsymbol{\beta}_j$ is the related parameter vector.

Algorithm 1 FSS With Forward Selection Process

1: Give the training dataset $\boldsymbol{Y}$ with N samples and the reconstruction error threshold ε_{th};
2: **initialization** $\boldsymbol{Y}_S \leftarrow \emptyset$, $i \leftarrow 0$, set $\varepsilon_{\min}$ to a large positive number;
3: **while** $\varepsilon_{\min} \ge \varepsilon_{th}$ **do**
4: $\quad i = i + 1$;
5: $\quad$ **for** $1 \le n \le N - i + 1$ and $\boldsymbol{y}_n \in \boldsymbol{Y}$ **do**
6: $\quad\quad \boldsymbol{Y}_S^{[n]} \leftarrow \boldsymbol{Y}_S \bigcup \boldsymbol{y}_n$;
7: $\quad\quad$ Compute the overall error $\varepsilon^{[n]}$ for $\boldsymbol{Y}_S^{[n]}$;
8: $\quad$ **end for**
9: $\quad i_n = \arg\min_{1 \le n \le N-i+1} \varepsilon^{[n]}$;
10: $\quad \boldsymbol{Y}_S \leftarrow \boldsymbol{Y}_S \bigcup \boldsymbol{y}_{i_n}$;
11: $\quad \boldsymbol{Y} \leftarrow \boldsymbol{Y} \setminus \boldsymbol{y}_{i_n}$;
12: $\quad \varepsilon_{\min} = \varepsilon^{[i_n]}$;
13: **end while**

The feature samples $\boldsymbol{Y}_S$ are selected by minimizing the normalized error function ε_j as [45]

$$\varepsilon_j = \frac{\|\boldsymbol{y}_j - \widehat{\boldsymbol{y}}_j\|^2}{\|\boldsymbol{y}_j\|^2} = \frac{\left\|\boldsymbol{y}_j - \boldsymbol{Y}_S^T\boldsymbol{\beta}_j\right\|^2}{\|\boldsymbol{y}_j\|^2}. \tag{29}$$

For the given $\boldsymbol{Y}_S$, $\boldsymbol{\beta}_j$ is obtained as [45]

$$\boldsymbol{\beta}_j = \left(\boldsymbol{Y}_S\boldsymbol{Y}_S^T\right)^{-1}\boldsymbol{Y}_S\boldsymbol{y}_j. \tag{30}$$

By substituting (30) into (29), the optimization objective is reformulated as

$$\varepsilon_j = 1 - \frac{1}{\|\boldsymbol{y}_j\|^2}\boldsymbol{y}_j^T\boldsymbol{Y}_S^T\left(\boldsymbol{Y}_S\boldsymbol{Y}_S^T\right)^{-1}\boldsymbol{Y}_S\boldsymbol{y}_j. \tag{31}$$

Define $\boldsymbol{k}_{Sj} = \boldsymbol{Y}_S\boldsymbol{y}_j$, $\boldsymbol{K}_{SS} = \boldsymbol{Y}_S\boldsymbol{Y}_S^T$, and $k_{j,j} = \boldsymbol{y}_j^T\boldsymbol{y}_j = \|\boldsymbol{y}_j\|^2$. Then, (31) can be rewritten as

$$\varepsilon_j = 1 - \frac{\boldsymbol{k}_{Sj}^T\boldsymbol{K}_{SS}^{-1}\boldsymbol{k}_{Sj}}{k_{j,j}}. \tag{32}$$

Considering all the samples $\boldsymbol{y}_j$ for $1 \le j \le N$, the overall error is formulated as

$$\varepsilon = \frac{1}{N}\sum_{j=1}^{N}\varepsilon_j = 1 - \frac{1}{N}\sum_{j=1}^{N}\frac{\boldsymbol{k}_{Sj}^T\boldsymbol{K}_{SS}^{-1}\boldsymbol{k}_{Sj}}{k_{j,j}}. \tag{33}$$

The solution to the problem of minimizing ε can be obtained by a forward selection process [45], which is listed in Algorithm 1. The reconstruction error threshold ε_{th} trades off the reconstruction accuracy with the computational complexity, and its appropriate value is obviously problem-dependent. In this paper, we set the value of ε_{th} empirically.

The above FSS procedure is applied to select the feature-samples' subsets $\boldsymbol{T}_S^{(l-1)}$ for building the sparse kernel models in DePCA modeling. Thus, the DePCA optimization problem (24) can be reformulated as

$$\begin{aligned} &\max_{\boldsymbol{\alpha}_S^{(l)}} \frac{1}{N-1}\left(\boldsymbol{\alpha}_S^{(l)}\right)^T\boldsymbol{K}_{SN}^{(l-1)}\boldsymbol{K}_{NS}^{(l-1)}\boldsymbol{\alpha}_S^{(l)} \\ &\text{s.t.}\ \ \left(\boldsymbol{\alpha}_S^{(l)}\right)^T\boldsymbol{K}_{SS}^{(l-1)}\boldsymbol{\alpha}_S^{(l)} = 1 \end{aligned} \tag{34}$$

in which we have $\boldsymbol{K}_{SN}^{(l-1)} = \boldsymbol{\Phi}(\boldsymbol{T}_S^{(l-1)})\boldsymbol{\Phi}^T(\boldsymbol{T}^{(l-1)})$, $\boldsymbol{K}_{NS}^{(l-1)} = \boldsymbol{\Phi}(\boldsymbol{T}^{(l-1)})\boldsymbol{\Phi}^T(\boldsymbol{T}_S^{(l-1)})$, and $\boldsymbol{K}_{SS}^{(l-1)} = \boldsymbol{\Phi}(\boldsymbol{T}_S^{(l-1)})\boldsymbol{\Phi}^T(\boldsymbol{T}_S^{(l-1)})$. By solving the optimization (34), we obtain a set of the sparse eigenvectors $\boldsymbol{\alpha}_{Si}^{(l)} \in \mathbb{R}^{N_S^{(l-1)}}$ for $1 \le i \le \widetilde{N}_S^{(l)}$, where $\widetilde{N}_S^{(l)}$ is the number of the projection vectors $\boldsymbol{\alpha}_{Si}^{(l)}$ corresponding to the nonzero eigenvalues. Thus, the lth-layer testing features $\boldsymbol{t}_t^{(l)} = [t_{t,1}^{(l)}\, t_{t,2}^{(l)} \dots t_{t,\widetilde{N}_S^{(l)}}^{(l)}]^T$ can be computed based on the sparse kernel model as

$$t_{t,i}^{(l)} = \left(\boldsymbol{k}_{St}^{(l-1)}\right)^T \boldsymbol{\alpha}_{Si}^{(l)} \tag{35}$$

where $\boldsymbol{k}_{St}^{(l-1)} = [k_{St,1}^{(l-1)}\ k_{St,2}^{(l-1)} \dots k_{St,N_S^{(l-1)}}^{(l-1)}]^T = \boldsymbol{\Phi}(\boldsymbol{T}_S^{(l-1)})\boldsymbol{\Phi}(\boldsymbol{t}_t^{(l-1)}) \in \mathbb{R}^{N_S^{(l-1)}}$. By ensuring $N_S^{(l-1)} << N$, online sample monitoring time is reduced significantly.

C. Monitoring Statistics Based on Bayesian Inference

For DePCA-based online sample monitoring, each feature layer offers two monitoring statistics $T^{2(l)}$ and $Q^{(l)}$, where $1 \le l \le L$. Under the normal operation, all the monitoring statistics should be smaller than their confidence limits, denoted by $T_{\lim}^{2(l)}$ and $Q_{\lim}^{(l)}$. We determine the confidence limits using kernel density estimation (KDE), which has the advantage of needing no specific data distribution assumption and has been used in many process monitoring methods [30], [47], [49]. Specifically, the normal operation data are projected onto the statistical models and the monitoring statistics $T_{\lim}^{2(l)}$ and $Q_{\lim}^{(l)}$ are computed. Then, the density function is estimated for each statistic by the KDE method. With the specified significance level δ, the confidence limit is determined by finding the point occupying the $1-\delta$ area of the density function. In this paper, the significance level δ is set to 5% and the 95% confidence limit is applied for all statistics.

To integrate the monitoring statistics at all the feature layers, a fusion strategy based on Bayesian inference is proposed to generate an overall decision, indicating the process status. With this strategy, Bayesian inference is first applied to turn each monitoring statistic into a posterior fault probability. Then, all the probability values from different layers are weighted to construct the probability-based monitoring statistics.

For the monitoring statistics $T^{2(l)}$ and $Q^{(l)}$, we denote the occurrence probabilities of sample $\boldsymbol{x}_t$ under fault condition C_f as $P_{T^2}^{(l)}(\boldsymbol{x}_t|C_f)$ and $P_Q^{(l)}(\boldsymbol{x}_t|C_f)$, which are defined, respectively, by [31], [54], [55]

$$P_{T^2}^{(l)}(\boldsymbol{x}_t|C_f) = \exp\left(-\gamma\, T_{\lim}^{2(l)}/T^{2(l)}\right) \tag{36}$$

$$P_Q^{(l)}(\boldsymbol{x}_t|C_f) = \exp\left(-\gamma\, Q_{\lim}^{(l)}/Q^{(l)}\right) \tag{37}$$

while the occurrence probabilities of sample $\boldsymbol{x}_t$ under normal operation condition C_n are denoted as $P_{T^2}^{(l)}(\boldsymbol{x}_t|C_n)$ and $P_Q^{(l)}(\boldsymbol{x}_t|C_n)$, which are computed, respectively, by [31], [54], [55]

$$P_{T^2}^{(l)}(\boldsymbol{x}_t|C_n) = \exp\left(-\gamma\, T^{2(l)}/T_{\lim}^{2(l)}\right) \tag{38}$$

$$P_Q^{(l)}(\boldsymbol{x}_t|C_n) = \exp\left(-\gamma\, Q^{(l)}/Q_{\lim}^{(l)}\right) \tag{39}$$

where γ is a tuning parameter designed to decrease the sensitivity to data outliers, and we set $\gamma = 0.2$ empirically.

According to Bayesian inference [31], [54], the monitoring statistics $T^{2(l)}$ and $Q^{(l)}$ can be transformed into the posterior fault probabilities $P_{T^2}^{(l)}(C_f|\boldsymbol{x}_t)$ and $P_Q^{(l)}(C_f|\boldsymbol{x}_t)$ by

$$P_{T^2}^{(l)}(C_f|\boldsymbol{x}_t) = \frac{P_{T^2}^{(l)}(\boldsymbol{x}_t|C_f)P_{T^2}^{(l)}(C_f)}{P_{T^2}^{(l)}(\boldsymbol{x}_t)}, \quad 1 \le l \le L \tag{40}$$

$$P_Q^{(l)}(C_f|\boldsymbol{x}_t) = \frac{P_Q^{(l)}(\boldsymbol{x}_t|C_f)P_Q^{(l)}(C_f)}{P_Q^{(l)}(\boldsymbol{x}_t)}, \quad 1 \le l \le L \tag{41}$$

where $P_{T^2}^{(l)}(C_f)$ and $P_Q^{(l)}(C_f)$ are the prior fault probabilities equivalent to the significance level δ, while $P_{T^2}^{(l)}(\boldsymbol{x}_t)$ and $P_Q^{(l)}(\boldsymbol{x}_t)$ are the occurrence probabilities of $\boldsymbol{x}_t$ given by

$$P_{T^2}^{(l)}(\boldsymbol{x}_t) = P_{T^2}^{(l)}(\boldsymbol{x}_t|C_f)P_{T^2}^{(l)}(C_f) + P_{T^2}^{(l)}(\boldsymbol{x}_t|C_n)P_{T^2}^{(l)}(C_n) \tag{42}$$

$$P_Q^{(l)}(\boldsymbol{x}_t) = P_Q^{(l)}(\boldsymbol{x}_t|C_f)P_Q^{(l)}(C_f) + P_Q^{(l)}(\boldsymbol{x}_t|C_n)P_Q^{(l)}(C_n) \tag{43}$$

in which $P_{T^2}^{(l)}(C_n)$ and $P_Q^{(l)}(C_n)$ are the prior normal probabilities equivalent to the confidence level $1-\delta$.

Based on (40) and (41), two probability-based overall monitoring statistics PT^2 and PQ are constructed by weighting the posterior fault probabilities at each layer according to

$$PT^2 = \frac{\sum_{l=1}^{L} w_{T^2}^{(l)} P_{T^2}^{(l)}(C_f|\boldsymbol{x}_t)}{\sum_{l=1}^{L} w_{T^2}^{(l)}} = \sum_{l=1}^{L} \bar{w}_{T^2}^{(l)} P_{T^2}^{(l)}(C_f|\boldsymbol{x}_t) \tag{44}$$

$$PQ = \frac{\sum_{l=1}^{L} w_Q^{(l)} P_Q^{(l)}(C_f|\boldsymbol{x}_t)}{\sum_{l=1}^{L} w_Q^{(l)}} = \sum_{l=1}^{L} \bar{w}_Q^{(l)} P_Q^{(l)}(C_f|\boldsymbol{x}_t) \tag{45}$$

where $w_{T^2}^{(l)}$ and $w_Q^{(l)}$ are the weighting factors, while $\bar{w}_{T^2}^{(l)} = w_{T^2}^{(l)} / \sum_{l=1}^{L} w_{T^2}^{(l)}$ and $\bar{w}_Q^{(l)} = w_Q^{(l)} / \sum_{l=1}^{L} w_Q^{(l)}$ are the normalized weighting factors.

In order to highlight the fault alarming information, if some statistic indicates the occurrence of fault, it should be given with a large weighting factor. Otherwise, the statistic should have a small weighting factor. Therefore, we design the weighting according to

$$w_{T^2}^{(l)} = \begin{cases} \dfrac{1}{\epsilon}, & \text{if } P_{T^2}^{(l)}(C_f|\boldsymbol{x}_t) > \delta \,\&\, \bar{P}_{T^2}^{(l)}(C_f|\boldsymbol{x}_t) > \delta \\ \epsilon, & \text{otherwise} \end{cases} \tag{46}$$

$$w_Q^{(l)} = \begin{cases} \dfrac{1}{\epsilon}, & \text{if } P_Q^{(l)}(C_f|\boldsymbol{x}_t) > \delta \,\&\, \bar{P}_Q^{(l)}(C_f|\boldsymbol{x}_t) > \delta \\ \epsilon, & \text{otherwise} \end{cases} \tag{47}$$

where $0 < \epsilon < 1$ is a very small number, and $\bar{P}_{T^2}^{(l)}(C_f|\boldsymbol{x}_t)$ and $\bar{P}_Q^{(l)}(C_f|\boldsymbol{x}_t)$ are the mean posterior fault probabilities over the past H testing samples, given by

$$\bar{P}_{T^2}^{(l)}(C_f|\boldsymbol{x}_t) = \frac{1}{H} \sum_{i=t-H+1}^{t} P_{T^2}^{(l)}(C_f|\boldsymbol{x}_i) \tag{48}$$

$$\bar{P}_Q^{(l)}(C_f|\boldsymbol{x}_t) = \frac{1}{H} \sum_{i=t-H+1}^{t} P_Q^{(l)}(C_f|\boldsymbol{x}_i). \tag{49}$$

In (46) and (47), $P_{T^2}^{(l)}(C_f|\boldsymbol{x}_t) > \delta$ and $P_Q^{(l)}(C_f|\boldsymbol{x}_t) > \delta$ are the judging condition related to the current sample, while $\bar{P}_{T^2}^{(c)}(C_f|\boldsymbol{x}_t) > \delta$ and $\bar{P}_Q^{(c)}(C_f|\boldsymbol{x}_t) > \delta$ are the judging condition based on the past H samples. When both these two conditions indicate a fault, the weighting factor is given a large value of $1/\epsilon$. Otherwise, the process is considered under the normal operation condition and a small weighting factor ϵ is assigned for the corresponding statistic.

Remark 2: The choice of H is to balance the conflicting requirements of achieving a high fault detection rate (FDR) while maintaining a low false alarming rate (FAR). This is because a large H value helps to decrease the influence of the noise in data, which leads to a low FAR. But a too large H is unable to reflect the current process status properly, which decreases the FDR. Similarly, for a relatively large ϵ value, fault information cannot be highlighted, which leads to a low FDR, although the FAR will also be low. For a sufficiently small ϵ, fault information can be properly highlighted, which achieves a high FDR, although the FAR will also be high. Therefore, appropriate values of H and ϵ must be determined empirically. In our case studies, the value of H and ϵ are empirically selected as $H = 6$ and $\epsilon = 0.01$.

With the aid of the weightings (46) and (47), the two probability-based overall monitoring statistics, (44) and (45), are built, which integrate the results of all the feature layers to indicate the process operating status. Specifically, if $PT^2 \leq \delta$ and $PQ \leq \delta$, the process is under the normal operation; otherwise, a fault sample is detected.

D. Process Monitoring Procedure Based on DePCA

The DePCA-based process monitoring procedure involves offline model training and online sample monitoring stages.

In the offline model training stage, normal operating data are collected and divided into training and validating data sets. The training data set is used to build the DePCA model and the validating data set is used to compute the confidence limits.

Offline Model Training Stage:

1) Collect the normal operation data set, divide it into the training data set $\boldsymbol{X}_{\text{tr}}$ and the validating data set $\boldsymbol{X}_{\text{va}}$, and normalize both the data sets with the mean and variance of the normal training data set.
2) Build the DePCA model using the training data $\boldsymbol{X}_{\text{tr}}$.
3) Project the validating data $\boldsymbol{X}_{\text{va}}$ onto the DePCA model and compute the monitoring statistics $T^{2(l)}$ and $Q^{(l)}$ for each layer.
4) Obtain the confidence limits $T_{\text{lim}}^{2(l)}$ and $Q_{\text{lim}}^{(l)}$ using the KDE method.

During the online monitoring stage, newly measured data sample is projected onto the DePCA model. The data features of the new sample are extracted and the monitoring statistics PT^2 and PQ are obtained to judge the process status.

Online Sample Monitoring Stage:

1) Acquire the new data $\boldsymbol{x}_t$ at the sample time t and normalize it with the mean and variance of the normal training data set.
2) Project $\boldsymbol{x}_t$ onto the DePCA model and compute the features of different layers.
3) Compute the layerwise monitoring statistics corresponding to $\boldsymbol{x}_t$ and obtain the overall probability-based monitoring statistics PT^2 and PQ using (44) and (45).
4) Judge if a fault occurs by comparing PT^2 and PQ with their respective significance levels.

IV. Case Studies

We conduct two case studies involving a simulated nonlinear system and the benchmark Tennessee Eastman (TE) process to verify the effectiveness of our proposed DePCA-based process monitoring approach, specifically, DePCA-N1 and DePCA-N2, in comparison with the KPCA- and SPCA-based methods. For all these methods of building statistical models, the 95% confidence limits are computed for each statistic by the KDE method. In all monitoring charts, the 95% confidence limit is plotted with the dashed line, while the monitoring statistic is plotted with the solid curve. Two performance indices are used to evaluate the monitoring results, which are the FDR and the fault detection time (FDT). The FDR is defined as the percentage of the samples exceeding the confidence limits over all the faulty samples, while the FDT is defined as the sample index of the alarming sample from which successive six samples exceed the confidence limit first time.

A. Simulated Nonlinear System

A nonlinear system with three variables is simulated according to the following model [26]:

$$\begin{cases} x_1 = u + e_1 \\ x_2 = u^2 - 3u + e_2 \\ x_3 = -u^3 + 3u^2 + e_3 \end{cases} \tag{50}$$

where x_1, x_2 and x_3 are the monitored variables and the source variable u obeys the uniform distribution in $[-2, 2]$, while the noises e_1, e_2 and e_3 follow the independent normal distributions with zero mean and variance 0.01. The normal operation data set of 1000 samples is generated based on (50). Five hundred samples of the normal operation data set are used as the training data to develop the statistical model and the other 500 samples are used as the validating data set to obtain the confidence limits. In order to test the fault detection performance, one fault is simulated, which is a step measurement bias of 1.0 in x_2. This fault case includes 500 samples where the fault is introduced after the 200th sample.

The kernel modeling of KPCA, SPCA, and DePCA-N1 as well as the second kernel modeling layer of DePCA-N2 involve the use of the Gaussian kernel function with the kernel width set to $\sigma = 500m$ empirically, where m denotes the dimension of the input vector to the corresponding kernel mapping layer. For DePCA-N2, additionally, the first kernel modeling layer adopts the two-order polynomial kernel function with the parameters empirically set to $d_0 = m$ and $d_1 = 2$. The values of m for various methods are listed in Table I. The reconstruction threshold ϵ_{th} is empirically chosen to be 0.0001 for both DePCA-N1 and DePCA-N2.

The fault detection results obtained by the KPCA, SPCA, DePCA-N1, and DePCA-N2 are illustrated in Figs. 6–9,

TABLE I

INPUT DIMENSION TO KERNEL MAPPING LAYER. THE NUMBER OF MONITORED VARIABLES IS M, AND $\widetilde{N}_S^{(2)}$ IS THE NUMBER OF NONLINEAR FEATURES EXTRACTED BY THE 1st NONLINEAR FEATURE LAYER (L4)

KPCA	$m = M$
SPCA	$m = M$
DePCA-N1	$m = M$
DePCA-N2, 1st kernel mapping layer (L3)	$m = M$
DePCA-N2, 2nd kernel mapping layer (L5)	$m = \widetilde{N}_S^{(2)}$

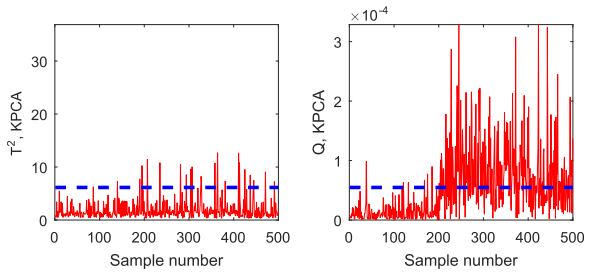


Fig. 6. KPCA monitoring charts for the simulated nonlinear system fault.

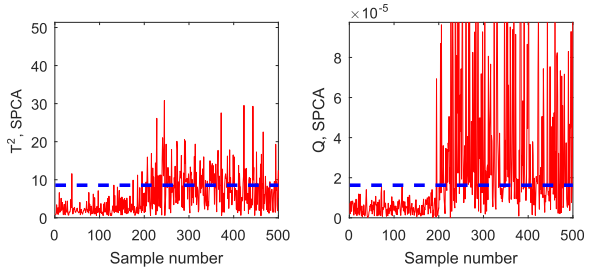


Fig. 7. SPCA monitoring charts for the simulated nonlinear system fault.

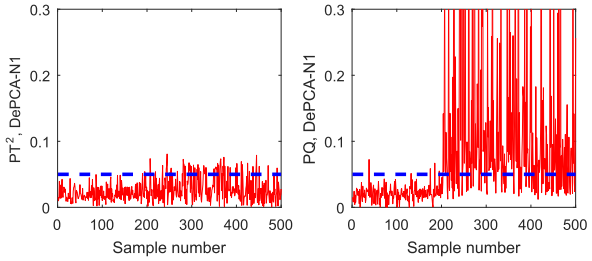


Fig. 8. DePCA-N1 monitoring charts for the simulated nonlinear system fault.

respectively. The results are also summarized in Table II, where "–" indicates that no successive six samples exceed the confidence limit, and thus the fault is not alarmed. According to Fig. 6, the KPCA T^2 statistic cannot detect the fault successfully, while its Q statistic detects the fault at the 240th sample. The FDRs of the KPCA T^2 and Q statistics are 6.0% and 58.3%, respectively. The SPCA method does better as can be seen from Fig. 7. Specifically, the SPCA T^2 and Q statistics obtain the FDRs of 41.0% and 61.0%, respectively, while its T^2 and Q statistics achieve the FDTs at the 240th and 219th samples, respectively. It can be seen from Fig. 8 that the DePCA-N1 PT^2 and PQ statistics obtain the FDRs

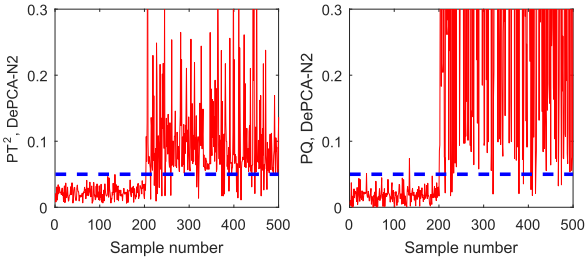


Fig. 9. DePCA-N2 monitoring charts for the simulated nonlinear system fault.

TABLE II

FDRS (%) AND FDTS (SAMPLE NUMBER) FOR THE SIMULATED NONLINEAR SYSTEM FAULT OBTAINED BY PCA, KPCA, AND DEPCA

Method	KPCA		SPCA		DePCA-N1		DePCA-N2	
Statistic	T^2	Q	T^2	Q	PT^2	PQ	PT^2	PQ
FDR	6.0	58.3	41.0	61.0	23.3	74.0	91.7	96.3
FDT	–	240	240	219	326	204	204	201

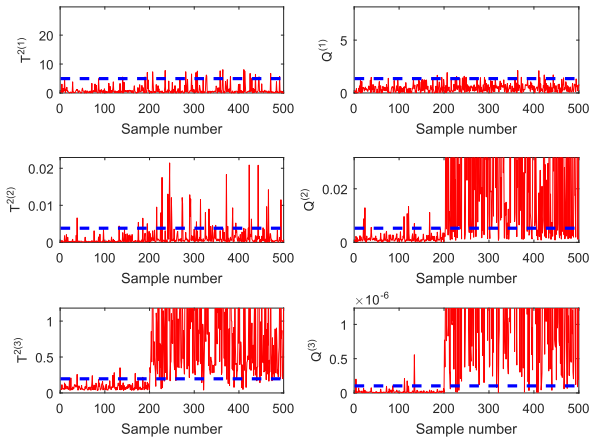


Fig. 10. DePCA-N2 monitoring statistics at each feature layer.

of 23.3% and 74.0%, respectively. Although the DePCA-N1 PT^2 statistic performs worse than the SPCA T^2 statistic, its PQ statistic detects the fault at the 204th sample, earlier than the SPCA method, while its PQ statistic also achieves a higher FDR than the SPCA. With a deeper three-feature-layer structure, the DePCA-N2 obtains the best monitoring performance whose FDRs are 91.7% and 96.3% for the PT^2 and PQ statistics, respectively, while whose FDTs are the 204th and 201th samples for the two statistics, respectively.

To provide insight on how the deep model structure improves fault detection performance, we analyze the DePCA-N2 monitoring statistics at each feature layer in Fig. 10. It can be observed that the fault detection capability is enhanced layer by layer. The normalized weighting factors are displayed in Fig. 11. It is clear that all the statistics have the same weights before the fault is introduced at the 200th sample. When the fault occurs, the statistics that can indicate the occurrence of the fault have the bigger weight values. Observe that since the L2 linear feature-based statistics fail to sound the alarm, the weighting values for most of the faulty samples fall to zero. The L4 nonlinear feature-based statistics become

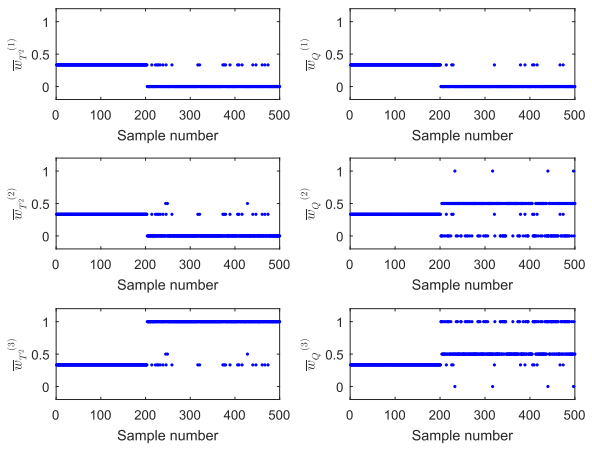


Fig. 11. Weighting factors for the layerwise monitoring statistics of DePCA-N2.

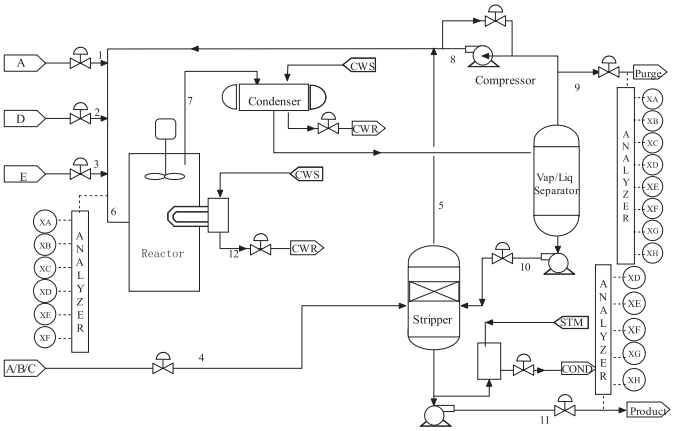


Fig. 12. TE process flowchart.

capable of detecting the fault, and many of the faulty samples are assigned with nonzero weight values. Since the fault detection capability of the L6 nonlinear feature-based statistics is significantly enhanced, most of the faulty samples are assigned with large nonzero weight values and many of these weights take the largest value of 1. Evidently, deep feature extraction is beneficial to detect process faults and our weighting strategy is effective to highlight the fault information.

B. TE Process

The TE process, detailed by Downs and Vogel [56], has been a benchmark process for evaluating the process monitoring and fault diagnosis methods [21], [30], [31], [54], [57]. The TE process is a simulated real chemical process, whose flowchart is illustrated in Fig. 12, and it involves five major units: reactor, condenser, compressor, stripper, and separator. For process monitoring modeling, we select 52 variables as monitored variables, including 22 continuous process variables, 19 composition measurement variables, and 11 manipulated variables. Normal operation and many fault cases have been designed for this process. The 21 faults used in our study are listed in Table III. The corresponding simulation data can be downloaded from

TABLE III
LIST OF THE TE PROCESS FAULTS USED

Fault label	Fault variable	Fault type
F1	A/C feed ratio(stream 4)	Step
F2	B composition(stream 4)	Step
F3	D feed temperature (stream 2)	Step
F4	Reactor cooling water inlet temperature	Step
F5	Condenser cooling water inlet temperature	Step
F6	A feed loss (stream 1)	Step
F7	C header pressure loss-reduced availability	Random
F8	A,B,and C feed compositions (stream 4)	Random
F9	D feed temperature (stream 2)	Random
F10	C feed temperature (Stream 4)	Random
F11	Reactor cooling water inlet temperature	Random
F12	Condenser cooling water inlet temperature	Random
F13	Reaction kinetics	Slow drift
F14	Reactor cooling water valve	Sticking
F15	Condenser cooling water valve	Sticking
F16-20	Unknown	Unknown
F21	Stream 4 valve	Sticking

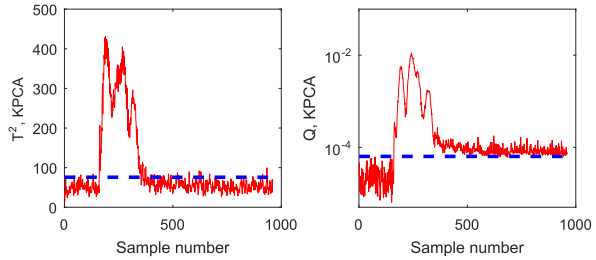


Fig. 13. KPCA monitoring charts for the TE process fault F5.

http://web.mit.edu/braatzgroup/links.html. The normal operation data include 1460 samples, and we use 500 samples to construct the training data set and the other 960 samples as validating data set. Each fault data set involves 960 samples and the fault is introduced after the 160th sample.

For the KPCA, SPCA, and DePCA-N1 as well as the second kernel mapping layer of DePCA-N2, the Gaussian kernel function is used with a kernel width of $\sigma = 500m$ found empirically. For the first kernel mapping layer of DePCA-N2, the two-order polynomial kernel function is adopted with the parameters $d_0 = 100m$ and $d_1 = 2$. The reconstruction error threshold ε_{th} for the FSS-based sparse modeling in DePCA-N1 and DePCA-N2 is empirically set to 0.002. The number of the retained linear or nonlinear PCs is determined by the average eigenvalue method. The confidence limits of the monitoring statistics are computed by the KDE method.

We first use the fault F5 to compare the fault detection performance of various methods. This fault is related to the step change in the condenser cooling water inlet temperature. When this fault occurs, the close-loop controller will compensate its influence, and as a result, the fault seems to disappear after a period of time. In fact, the fault is still existing but "is hidden" by feedback control. The KPCA monitoring charts for this fault are shown in Fig. 13, where it can be seen that both the T^2 and Q statistics detect this fault at the 161th sample. However, after the 350th sample, both the statistics decrease sharply, with the T^2 statistic under the confidence

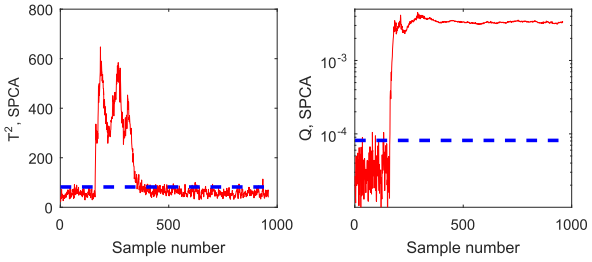


Fig. 14. SPCA monitoring charts for the TE process fault F5.

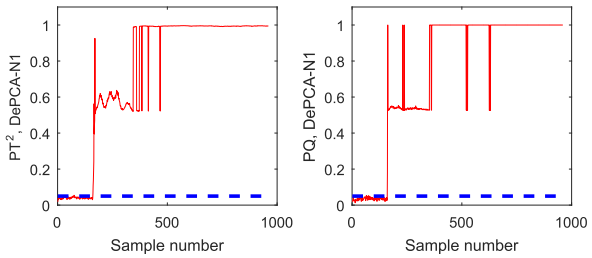


Fig. 15. DePCA-N1 monitoring charts for the TE process fault F5.

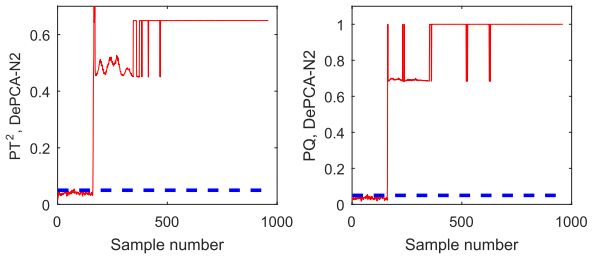


Fig. 16. DePCA-N2 monitoring charts for the TE process fault F5.

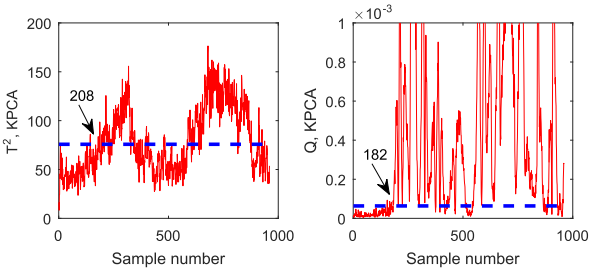


Fig. 17. KPCA monitoring charts for the TE process fault F10.

limit and the Q statistic just above the confidence limit. Hence, the KPCA monitoring charts may mislead the operator to believe that the fault has gone. From the SPCA monitoring charts shown in Fig. 14, we observe that its T^2 statistic performs very similarly to the KPCA's T^2 statistic, but its Q statistic does much better, which clearly and consistently exceeds the confidence limit. Figs. 15 and 16 confirm that both the DePCA-N1 and DePCA-N2 outperform the SPCA. Specifically, the two DePCA monitoring statistics exceed the confidence limit with 100% FDR during the faulty period.

Next, we investigate the monitoring performance of various methods on the fault F10, which involves the random variations of stream 4 feed temperature. The monitoring charts from the four methods are depicted in Figs. 17–20, respectively.

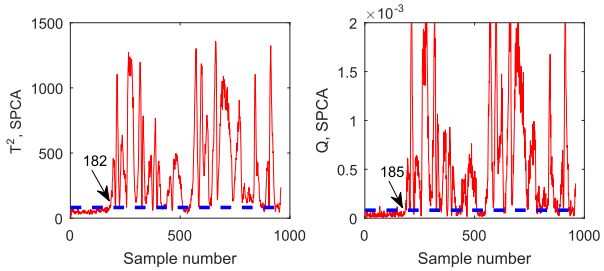


Fig. 18. SPCA monitoring charts for the TE process fault F10.

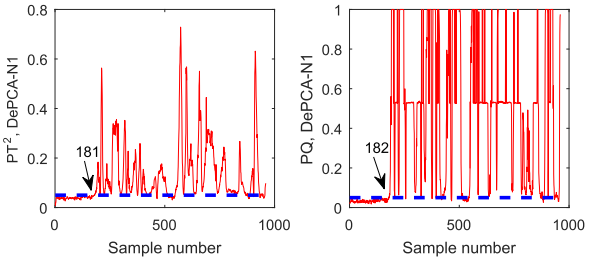


Fig. 19. DePCA-N1 monitoring charts for the TE process fault F10.

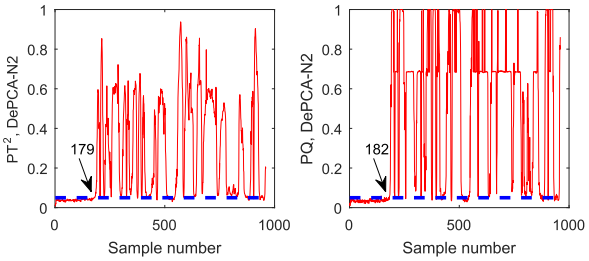


Fig. 20. DePCA-N2 monitoring charts for the TE process fault F10.

It can be seen from Fig. 17 that the KPCA T^2 and Q statistics detect this fault at the 208th and 182th samples, respectively, and the corresponding FDRs are 54.9% and 86.9%. According to Fig. 18, the SPCA T^2 and Q statistics alarm this fault at the 182th sample and the 185th sample, respectively, with the FDRs of 89.5% and 82.1%. As shown in Fig. 19, the DePCA-N1 PT^2 statistic detects the fault at the 181th sample and achieves the FDR of 89.0%, which is similar to the SPCA T^2 statistic. However, the DePCA-N1 PQ statistic detects the fault at the 182th sample and has an 89.9% FDR, which is better than the SPCA Q statistic. The monitoring results of the DePCA-N2 plotted in Fig. 20 indicate that its PT^2 statistic detects the fault at the 179th sample while its PQ statistic gives the fault alarm at the 182th sample.

Table IV summarizes the FDRs obtained by the KPCA, SPCA, DePCA-N1, and DePCA-N2 methods for all the 21 TE process faults. Observe that the DePCA-N1 attains a higher average FDR than the KPCA and SPCA. However, the monitoring results of the DePCA-N1 and DePCA-N2 are very close. This may be because for these particular fault data, the DePCA-N1 has extracted enough feature information and the DePCA-N2 could hardly enhance it with one more nonlinear feature layer. Further examining the results of Table IV, we observe that the two DePCA methods achieve similar

TABLE IV

FDRs (%) OF THE 21 TESTED TE PROCESS FAULTS OBTAINED BY THE KPCA, SPCA, AND DePCA METHODS

Fault label	KPCA		SPCA		DePCA-N1		DePCA-N2	
	T^2	Q	T^2	Q	PT^2	PQ	PT^2	PQ
F1	99.8	99.8	99.9	99.8	99.6	99.8	99.8	99.8
F2	98.8	98.5	98.9	99.0	98.6	98.8	98.8	98.8
F3	8.0	7.5	7.9	7.4	6.0	7.4	6.9	4.9
F4	**100**	**37.3**	**100**	**94.0**	**100**	**100**	**100**	**100**
F5	**28.6**	**99.5**	**30.5**	**99.9**	**100**	**100**	**100**	**100**
F6	99.5	100	100	100	100	100	100	100
F7	100	99.9	100	100	100	100	100	100
F8	98.3	98.1	98.1	97.9	98.0	98.1	98.0	98.1
F9	6.5	4.5	7.1	6.0	4.3	7.5	4.9	3.5
F10	**54.9**	**86.9**	**89.5**	**82.1**	**89.0**	**89.9**	**91.0**	**89.9**
F11	**79.3**	**51.8**	**79.8**	**64.8**	**75.4**	**77.5**	**77.8**	**77.5**
F12	99.1	99.5	99.5	99.8	99.9	99.9	99.9	99.9
F13	95.5	95.9	95.6	95.4	95.5	95.5	95.5	95.4
F14	100	99.9	100	100	99.9	100	100	100
F15	**13.3**	**14.0**	**15.9**	**10.8**	**15.5**	**17.9**	**19.1**	**14.4**
F16	**37.0**	**90.0**	**93.0**	**75.8**	**91.4**	**93.3**	**94.0**	**93.5**
F17	**96.1**	**90.5**	**96.5**	**92.3**	**94.9**	**96.5**	**96.5**	**96.4**
F18	91.3	89.4	91.1	90.4	90.6	91.0	90.4	90.8
F19	**19.1**	**80.8**	**75.0**	**90.4**	**87.3**	**89.8**	**91.1**	**91.4**
F20	**68.4**	**72.3**	**73.5**	**82.6**	**90.1**	**90.6**	**90.6**	**90.6**
F21	**54.5**	**44.3**	**56.5**	**59.0**	**60.4**	**59.5**	**62.0**	**57.5**
Mean	68.9	74.3	76.6	78.4	80.8	81.6	81.7	81.1

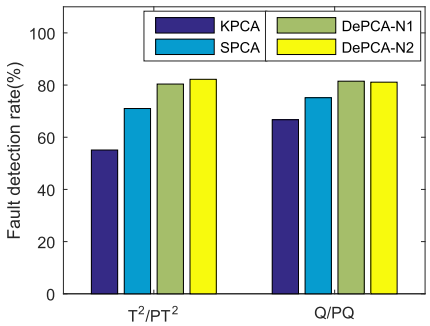


Fig. 21. Average FDRs obtained by the four methods over the faults F4, F5, F10, F11, F15, F16, F17, F19, F20, and F21.

performance as the KPCA and SPCA methods on the faults F1, F2, F3, F6, F7, F8, F9, F12, F13, F14, and F18, while the DePCA-N1 and DePCA-N2 outperform the KPCA and SPCA when detecting the faults F4, F5, F10, F11, F15, F16, F17, F19, F20, and F21. Fig. 21 shows the average FDRs over these 10 faults obtained by the four methods. It can be seen from Table IV that all the four methods fail for the faults F3, F9, and F15. In fact, it is well known that these three faults are extremely difficult for data-driven monitoring methods owing to the reason that there exist no observable changes in the mean or variance of these three fault data sets [58].

To investigate the influence of the kernel width on the achievable performance, in Table V, we list the average FDRs achieved by the four methods, given different Gaussian kernel widths of $\sigma = 50m$, $100m$, $500m$, and $1000m$. For the first kernel mapping layer of DePCA-N2, the parameter d_0 is fixed to $100m$. From Table V, we observe that for the KPCA, SPCA, and DePCA-N1, the achievable performance are influenced by the value of σ used, and $\sigma = 500m$ seems to be an appropriate choice. In contrast, for the DePCA-N2,

TABLE V

AVERAGE FDRs (%) OF THE 21 TESTED TE PROCESS FAULTS OBTAINED BY THE KPCA, SPCA, AND DePCA METHODS, GIVEN DIFFERENT GAUSSIAN KERNEL WIDTHS. THE PARAMETER d_0 IS FIXED TO 100 m FOR THE FIRST KERNEL MAPPING LAYER OF DePCA-N2

Kernel width σ	KPCA		SPCA		DePCA-N1		DePCA-N2	
	T^2	Q	T^2	Q	PT^2	PQ	PT^2	PQ
$50m$	59.3	69.3	76.1	77.9	77.7	82.1	81.8	81.4
$100m$	61.7	70.8	76.3	78.5	77.7	81.9	81.9	81.3
$500m$	68.9	74.3	76.6	78.4	80.8	81.6	81.7	81.1
$1000m$	69.6	73.5	76.6	76.2	80.1	81.9	81.7	80.9

TABLE VI

AVERAGE FARs (%) OF THE TESTED TE PROCESS DATA SETS OBTAINED BY THE KPCA, SPCA, AND DePCA METHODS

Method	KPCA		SPCA		DePCA-N1		DePCA-N2	
Statistic	T^2	Q	T^2	Q	PT^2	PQ	PT^2	PQ
Average FAR	2.8	2.7	3.0	4.6	2.3	3.0	2.8	1.2

the achievable performance remains consistently good for all the four choices of σ. We further fix $\sigma = 500m$ for the second kernel mapping layer of DePCA-N2 and vary its first kernel mapping layer parameter to $d_0 = 50m$, $100m$, $500m$, and $1000m$. Again, the achievable performance of the DePCA-N2 remains consistent with that of Table V. Intriguingly, the DePCA-N2 is more robust to its kernel parameters. Thus, even though the DePCA-N2 has more kernel mapping layers, it is in fact easier to choose its kernel parameters appropriately.

The FAR, defined as the percentage of the samples exceeding the confidence limit under the normal operation condition, is also an important metric in process monitoring. As the 95% confidence limit is used as the fault detection threshold, up to 5% of the normal samples may exceed the confidence limit statistically. That is, the FAR should not exceed 5%. The first 160 samples of each fault data set are the normal operation samples, which are used to calculate the FAR, and the average FAR is obtained by averaging over all the 3360 normal samples of the 21 fault data sets. Table VI lists the average FARs of the four methods, where it can be seen that all the average FARs are lower than 5%, and the DePCA-N2 achieves lower average FARs than the KPCA, SPCA, and DePCA-N1.

To compare the computation complexities of the KPCA, SPCA, DePCA-N1, and DePCA-N2, we list their main computation tasks for monitoring single online sample in Table VII. It is easy to understand that the KPCA requires less computation tasks than the SPCA and DePCA methods, because the KPCA is the shallowest nonlinear modeling. Also the online computational complexity of the SPCA should be very close to that of the KPCA, because the linear feature extraction requires very little extra computational cost. Furthermore, the complexity of the DePCA-N1 may actually be lower than the SPCA, as the FSS-based sparse kernel model projection may extract a smaller number of nonlinear features. Finally, it can be seen that the complexity of the DePCA-N2 should be around twice of the DePCA-N1. To verify these analyses, we evaluate the online computation time (C-Time) of each method. Specifically, we run the online monitoring programs

TABLE VII
COMPUTATION TASKS AND C-TIMES FOR MONITORING SINGLE ONLINE SAMPLE BY THE KPCA, SPCA, AND DEPCA METHODS

Method	Main computation tasks per sample	Average C-time per sample (s)
KPCA	Kernel model projection Nonlinear monitoring statistics	9.35×10^{-3}
SPCA	Linear model projection Kernel model projection Hybrid monitoring statistics	9.54×10^{-3}
DePCA-N1	Linear model projection Linear monitoring statistics Sparse kernel model projection Sparse kernel monitoring statistics Overall monitoring statistics	7.82×10^{-3}
DePCA-N2	Linear model projection Linear monitoring statistics 1st sparse kernel model projection 1st sparse kernel monitoring statistics 2nd sparse kernel model projection 2nd sparse kernel monitoring statistics Overall monitoring statistics	1.82×10^{-2}

TABLE VIII
AVERAGE C-TIMES (S) FOR MONITORING SINGLE ONLINE SAMPLE BY THE DEPCA METHODS WITH AND WITHOUT THE FSS

Method	without FSS	with FSS	saving
DePCA-N1	9.62×10^{-3}	7.82×10^{-3}	19%
DePCA-N2	2.30×10^{-2}	1.82×10^{-2}	21%

of different methods 10 times on the same computer, configured with Intel Core i7-5500U processor (2.4 GHz) and 8G RAM memory. The average C-Times per sample of the four methods are also listed in Table VII. The results of Table VII clearly confirm our computational complexity analysis.

To validate the effectiveness of the FSS method, we run the online monitoring programs of the DePCA algorithms with the FSS and without the FSS on the same computational platform, given in the previous paragraph, and the online computational times obtained are compared in Table VIII. It can be seen that for the DePCA-N1 structure, using the FSS method, yields approximately 19% saving in the online computational time, while for the DePCA-N2 structure, saving in the online computational time is about 21%. Furthermore, it can be seen by comparing Table VII with Table VIII that the online computational time required by the DePCA-N1 without employing the FSS is similar to that of the SPCA, which is as expected.

V. CONCLUSION

A novel layerwise feature extraction-based deep statistical modeling approach, referred to as DePCA, has been proposed for nonlinear process monitoring. Our contribution has been threefold. First and most importantly, we have presented a deep PCA model structure by introducing deep learning to improve the existing nonlinear PCA methods. Our DePCA applies PCA and KPCA to extract the multiple layers of linear and nonlinear features efficiently, and it does not require complicated nonlinear neural network optimization. Under this framework, two practical DePCA-N1 and DePCA-N2 models have been adopted for process monitoring. Second, an FSS technique has been applied to build a sparse kernel model of DePCA, which is capable of significantly reducing the computation loads caused by multiple layers of feature extraction. Third, a Bayesian inference-based monitoring strategy has been constructed to fuse all the monitoring statistics at different feature layers. Simulation results including a simulated nonlinear system and the benchmark TE process have validated the effectiveness of the DePCA method over the existing state-of-the-art KPCA methods.

This paper has opened up a new research direction for KPCA-based nonlinear process monitoring by exploiting hierarchical feature extraction. There are many theoretical and practical issues deserving further investigation. An important question is how "deep" should be, or how many feature layers are enough to mine the latent data features sufficiently for a given problem. Further research is warranted to study this issue. This paper focuses on fault detection. After a fault is detected, it is necessary to carry out fault identification in order to find out the root cause of the fault. How to realize the fault identification in DePCA-based process monitoring is worthy further investigating.

REFERENCES

[1] S. Yin, X. Li, H. Gao, and O. Kaynak, "Data-based techniques focused on modern industry: An overview," *IEEE Trans. Ind. Electron.*, vol. 62, no. 1, pp. 657–667, Jan. 2015.
[2] Y. Zhang, T. Chai, Z. Li, and C. Yang, "Modeling and monitoring of dynamic processes," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 2, pp. 277–284, Feb. 2012.
[3] S. J. Qin, "Survey on data-driven industrial process monitoring and diagnosis," *Annu. Rev. Control*, vol. 36, no. 2, pp. 220–234, Dec. 2012.
[4] Z. Ge, "Review on data-driven modeling and monitoring for plant-wide industrial processes," *Chemometrics Intell. Lab. Syst.*, vol. 171, pp. 16–25, Dec. 2017.
[5] Z. Ge, Z. Song, S. X. Ding, and B. Huang, "Data mining and analytics in the process industry: The role of machine learning," *IEEE Access*, vol. 5, pp. 20590–20616, 2017.
[6] L. I. Kuncheva and W. J. Faithfull, "PCA feature extraction for change detection in multidimensional unlabeled data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 1, pp. 69–80, Jan. 2014.
[7] J. Zhu, Z. Ge, and Z. Song, "HMM-driven robust probabilistic principal component analyzer for dynamic process fault classification," *IEEE Trans. Ind. Electron.*, vol. 62, no. 6, pp. 3814–3821, Jun. 2015.
[8] Y. Xu and X. Deng, "Fault detection of multimode non-Gaussian dynamic process using dynamic Bayesian independent component analysis," *Neurocomputing*, vol. 200, pp. 70–79, Aug. 2016.
[9] X. Tian, L. Cai, and S. Chen, "Noise-resistant joint diagonalization independent component analysis based process fault detection," *Neurocomputing*, vol. 149, pp. 652–666, Feb. 2015.
[10] S. Moon and H. Qi, "Hybrid dimensionality reduction method based on support vector machine and independent component analysis," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 5, pp. 749–761, May 2012.
[11] B. Jiang, X. Zhu, D. Huang, J. A. Paulson, and R. D. Braatz, "A combined canonical variate analysis and Fisher discriminant analysis (CVA–FDA) approach for fault diagnosis," *Comput. Chem. Eng.*, vol. 77, pp. 1–9, Jun. 2015.
[12] C. Ruiz-cárcel, L. Lao, Y. Cao, and D. Mba, "Canonical variate analysis for performance degradation under faulty conditions," *Control Eng. Pract.*, vol. 54, pp. 70–80, Sep. 2016.
[13] S. Yin, S. X. Ding, A. Haghani, H. Hao, and P. Zhang, "A comparison study of basic data-driven fault diagnosis and process monitoring methods on the benchmark Tennessee Eastman process," *J. Process Control*, vol. 22, no. 9, pp. 1567–1581, 2012.
[14] J. Feng, J. Wang, H. Zhang, and Z. Han, "Fault diagnosis method of joint fisher discriminant analysis based on the local and global manifold learning and its kernel version," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 1, pp. 122–133, Jan. 2016.

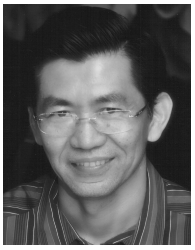
[15] K. Peng, K. Zhang, B. You, J. Dong, and Z. Wang, "A quality-based nonlinear fault diagnosis framework focusing on industrial multimode batch processes," *IEEE Trans. Ind. Electron.*, vol. 63, no. 4, pp. 2615–2624, Apr. 2016.
[16] Q. Liu, S. J. Qin, and T. Chai, "Multiblock concurrent PLS for decentralized monitoring of continuous annealing processes," *IEEE Trans. Ind. Electron.*, vol. 61, no. 11, pp. 6429–6437, Nov. 2014.
[17] W. Ku, R. H. Storer, and C. Georgakis, "Disturbance detection and isolation by dynamic principal component analysis," *Chemometrics Intell. Lab. Syst.*, vol. 30, no. 1, pp. 179–196, 1995.
[18] G. Li, S. J. Qin, and D. Zhou, "A new method of dynamic latent-variable modeling for process monitoring," *IEEE Trans. Ind. Electron.*, vol. 61, no. 11, pp. 6438–6445, Nov. 2014.
[19] I. Portnoy, K. Melendez, H. Pinzon, and M. Sanjuan, "An improved weighted recursive PCA algorithm for adaptive fault detection," *Control Eng. Pract.*, vol. 50, pp. 69–83, May 2016.
[20] Q. Jiang, B. Huang, and X. Yan, "GMM and optimal principal components-based Bayesian method for multimode fault diagnosis," *Comput. Chem. Eng.*, vol. 84, pp. 338–349, Jan. 2016.
[21] H. Ma, Y. Hu, and H. Shi, "A novel local neighborhood standardization strategy and its application in fault detection of multimode processes," *Chemometrics Intell. Lab. Syst.*, vol. 118, pp. 287–300, Aug. 2012.
[22] X. Deng and X. Tian, "Multimode process fault detection using local neighborhood similarity analysis," *Chin. J. Chem. Eng.*, vol. 22, nos. 11–12, pp. 1260–1267, Nov. 2014.
[23] Q. Liu, S. J. Qin, and T. Chai, "Decentralized fault diagnosis of continuous annealing processes based on multilevel PCA," *IEEE Trans. Autom. Sci. Eng.*, vol. 10, no. 3, pp. 687–698, Jul. 2013.
[24] C. Zhao and F. Gao, "Statistical modeling and online fault detection for multiphase batch processes with analysis of between-phase relative changes," *Chemometrics Intell. Lab. Syst.*, vol. 130, pp. 58–67, Jan. 2014.
[25] 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, 1998.
[26] J.-M. Lee, C. K. Yoo, S. W. Choi, P. A. Vanrolleghem, and I.-B. Lee, "Nonlinear process monitoring using kernel principal component analysis," *Chem. Eng. Sci.*, vol. 59, no. 1, pp. 223–234, Jan. 2004.
[27] S. W. Choi, C. K. Lee, J.-M. Lee, J. H. Park, and I.-B. Lee, "Fault detection and identification of nonlinear processes based on kernel PCA," *Chemometrics Intell. Lab. Syst.*, vol. 75, no. 1, pp. 55–67, Jan. 2005.
[28] Y. Zhang and C. Ma, "Fault diagnosis of nonlinear processes using multiscale KPCA and multiscale KPLS," *Chem. Eng. Sci.*, vol. 66, no. 1, pp. 64–72, 2011.
[29] J. Yi, D. Huang, S. Fu, H. He, and T. Li, "Optimized relative transformation matrix using bacterial foraging algorithm for process fault detection," *IEEE Trans. Ind. Electron.*, vol. 63, no. 4, pp. 2595–2605, Apr. 2016.
[30] X. Deng, X. Tian, and S. Chen, "Modified kernel principal component analysis based on local structure analysis and its application to nonlinear process fault diagnosis," *Chemometrics Intell. Lab. Syst.*, vol. 127, pp. 195–209, Aug. 2013.
[31] Q. Jiang and X. Yan, "Nonlinear plant-wide process monitoring using MI-spectral clustering and Bayesian inference-based multiblock KPCA," *J. Process Control*, vol. 32, pp. 38–50, Aug. 2015.
[32] C.-Y. Cheng, C.-C. Hsu, and M.-C. Chen, "Adaptive kernel principal component analysis (KPCA) for monitoring small disturbances of nonlinear processes," *Ind. Eng. Chem. Res.*, vol. 49, no. 5, pp. 2254–2262, 2010.
[33] L. Cai, X. Tian, and S. Chen, "Monitoring nonlinear and non-Gaussian processes using Gaussian mixture model-based weighted kernel independent component analysis," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 1, pp. 122–135, Jan. 2017.
[34] Z. Ge, C. Yang, and Z. Song, "Improved kernel PCA-based monitoring approach for nonlinear processes," *Chem. Eng. Sci.*, vol. 64, no. 9, pp. 2245–2255, May 2009.
[35] X. Deng and X. Tian, "Nonlinear process fault pattern recognition using statistics kernel PCA similarity factor," *Neurocomputing*, vol. 121, pp. 298–308, Dec. 2013.
[36] X. Deng, X. Tian, S. Chen, and C. J. Harris, "Nonlinear process fault diagnosis based on serial principal component analysis," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 3, pp. 560–572, Mar. 2018.
[37] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.
[38] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
[39] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.
[40] I. Arel, D. C. Rose, and T. P. Karnowski, "Deep machine learning—A new frontier in artificial intelligence research [research frontier]," *IEEE Comput. Intell. Mag.*, vol. 5, no. 4, pp. 13–18, Nov. 2010.
[41] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.
[42] S. Kim, B. Park, B. S. Song, and S. Yang, "Deep belief network based statistical feature learning for fingerprint liveness detection," *Pattern Recognit. Lett.*, vol. 77, pp. 58–65, Jul. 2016.
[43] K. Audhkhasi, O. Osoba, and B. Kosko, "Noise-enhanced convolutional neural networks," *Neural Netw.*, vol. 78, pp. 15–23, Jun. 2016.
[44] C. Xia, F. Qi, and G. Shi, "Bottom—Up visual saliency estimation with deep autoencoder-based sparse reconstruction," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 6, pp. 1227–1240, Jun. 2016.
[45] G. Baudat and F. Anouar, "Feature vector selection and projection using kernels," *Neurocomputing*, vol. 55, nos. 1–2, pp. 21–38, Sep. 2003.
[46] T. Kourti and J. F. MacGregor, "Process analysis, monitoring and diagnosis, using multivariate projection methods," *Chemometrics Intell. Lab. Syst.*, vol. 28, no. 1, pp. 3–21, 1995.
[47] X. Tian, X. Zhang, X. Deng, and S. Chen, "Multiway kernel independent component analysis based on feature samples for batch process monitoring," *Neurocomputing*, vol. 72, nos. 7–9, pp. 1584–1596, Mar. 2009.
[48] S. Chen, X. Hong, C. J. Harris, and P. M. Sharkey, "Sparse modeling using orthogonal forward regression with PRESS statistic and regularization," *IEEE Trans. Syst., Man, Cybern. B. Cybern.*, vol. 34, no. 2, pp. 898–911, Apr. 2004.
[49] S. Chen, X. Hong, and C. J. Harris, "Sparse kernel density construction using orthogonal forward regression with leave-one-out test score and local regularization," *IEEE Trans. Syst., Man, Cybern. B. Cybern.*, vol. 34, no. 4, pp. 1708–1717, Aug. 2004.
[50] S. Chen, X. Hong, B. L. Luk, and C. J. Harris, "Construction of tunable radial basis function networks using orthogonal forward selection," *IEEE Trans. Syst., Man, Cybern. B. Cybern.*, vol. 39, no. 2, pp. 457–466, Apr. 2009.
[51] S. Chen, X. Hong, and C. J. Harris, "Probability density estimation with tunable kernels using orthogonal forward regression," *IEEE Trans. Syst., Man, Cybern. B. Cybern.*, vol. 40, no. 4, pp. 1101–1114, Aug. 2010.
[52] X. Hong, S. Chen, J. Gao, and C. J. Harris, "Nonlinear identification using orthogonal forward regression with nested optimal regularization," *IEEE Trans. Cybern.*, vol. 45, no. 12, pp. 2925–2936, Dec. 2015.
[53] X. Deng and X. Tian, "Sparse kernel locality preserving projection and its application in nonlinear process fault detection," *Chin. J. Chem. Eng.*, vol. 21, no. 2, pp. 163–170, 2013.
[54] Z. Ge and Z. Song, "Performance-driven ensemble learning ICA model for improved non-Gaussian process monitoring," *Chemometrics Intell. Lab. Syst.*, vol. 123, pp. 1–8, Apr. 2013.
[55] Z. Ge, M. Zhang, and Z. Song, "Nonlinear process monitoring based on linear subspace and Bayesian inference," *J. Process Control*, vol. 20, no. 5, pp. 676–688, 2010.
[56] J. J. Downs and E. F. Vogel, "A plant-wide industrial process control problem," *Comput. Chem. Eng.*, vol. 17, no. 3, pp. 245–255, 1993.
[57] C. K. Lau, K. Ghosh, M. A. Hussain, and C. R. C. Hassan, "Fault diagnosis of Tennessee Eastman process with multi-scale PCA and ANFIS," *Chemometrics Intell. Lab. Syst.*, vol. 120, pp. 1–14, Jan. 2013.
[58] L. H. Chiang, E. L. Russell, and R. D. Braatz, *Fault Detection and Diagnosis in Industrial Systems*. London, U.K.: Springer-Verlag, 2001.

Xiaogang Deng received the B.Eng. and Ph.D. degrees from the China University of Petroleum, Dongying, China, in 2002 and 2008, respectively.

From 2015 to 2016, he was a Visiting Scholar with the Department of Electronics and Computer Sciences, University of Southampton, Southampton, U.K. He is currently an Associate Professor with the College of Information and Control Engineering, China University of Petroleum. His current research interests include industrial process modeling and simulation, data-driven fault detection and diagnosis, and control performance monitoring.

Xuemin Tian received the B.E. degree from the Huadong Petroleum Institute, Dongying, China, in 1982, and the M.S. degree from the Beijing University of Petroleum, Beijing, China, in 1994.

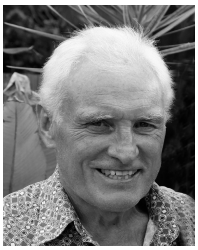
From 2001 to 2002, he was a Visiting Professor with Central of Process Control, University of California in Santa Barbara, Santa Barbara, CA, USA. He is currently a Professor of process control with the China University of Petroleum (Huadong), Dongying. His current research interests include modeling, advanced process control and optimization for petrochemical processes as well as fault detection and diagnosis, and process monitoring.



Sheng Chen (M'90–SM'97–F'08) received the B.Eng. degree in control engineering from the East China Petroleum Institute, Dongying, China, in 1982, the Ph.D. degree in control engineering from the City, University of London, London, U.K., in 1986, and the higher doctoral degree, Doctor of Sciences (D.Sc.) degree, from the University of Southampton, Southampton, U.K.

From 1986 to 1999, he held research and academic appointments with The University of Sheffield, Sheffield, U.K., The University of Edinburgh, Edinburgh, U.K., and The University of Portsmouth, Portsmouth, U.K. Since 1999, he has been with the School of Electronics and Computer Science, University of Southampton, Southampton, U.K., where he is currently a Professor of intelligent systems and signal processing. He is currently a Distinguished Adjunct Professor with King Abdulaziz University, Jeddah, Saudi Arabia. He has authored over 600 research papers. His current research interests include adaptive signal processing, wireless communications, modeling and identification of nonlinear systems, neural network and machine learning, intelligent control system design, evolutionary computation methods, and optimization.

Dr. Chen was an ISI highly cited researcher in engineering in 2004. He is a Fellow of the United Kingdom Royal Academy of Engineering and the IET.



Chris J. Harris received the B.Sc. degree from the University of Leicester, Leicester, U.K., the M.A. degree from the University of Oxford, Oxford, U.K., and the Ph.D. and D.Sc. degrees from the University of Southampton, Southampton, U.K., in 1972 and 2001, respectively.

He was the Deputy Chief Scientist of the U.K. Government. He is currently an Emeritus Research Professor with the University of Southampton, having previously held senior academic appointments at Imperial College, Oxford, and Manchester Universities. He has co-authored over 450 scientific research papers during a 45-year research career.

Dr. Harris was a recipient of the IEE senior Achievement Medal for data fusion research and the IEE Faraday Medal for distinguished international research in machine learning. He was elected to the U.K. Royal Academy of Engineering in 1996.