Deep learning based nonlinear principal component analysis for industrial process fault detection

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

Abstract—**Principal component analysis (PCA) and kernel PCA (KPCA) are the state-of-art machine learning methods widely used in industrial process monitoring and fault detection field. However, these methods build shallow statistical models based on single layer of features and may not achieve the best monitoring performance. In order to sufficiently mine the intrinsic data features, a deep learning based nonlinear PCA method, referred to as deep PCA (DePCA), is proposed in this paper. Motivated by the idea of deep learning, a layer-wise statistical model structure is designed to extract multi-layer data features, including both linear and nonlinear principal components. At each layer, two monitoring statistics are constructed to monitor the feature changes. For integrating the monitoring statistics of all feature layers, a Bayesian inference strategy is applied to convert the monitoring statistics into fault probabilities, which are weighted to form two probability-based comprehensive monitoring statistics for process fault detection. A case study using the benchmark Tennessee Eastman process demonstrates the superior performance of the proposed DePCA method over the traditional PCA and KPCA methods.**

I. Introduction

Machine learning plays an important role in our modern society. Some well-known machine learning algorithms such as principal component analysis (PCA), support vector machine (SVM) and artificial neural network (ANN) have achieved wide-ranging applications in many fields, including computer vision, web data mining, genomic medicine, wireless communication and industrial process monitoring [1]–[5]. Recently, a new powerful machine learning technique called deep learning has emerged and attracted significant attention from researchers. Different to traditional machine learning techniques with shallow model structure, deep learning systems usually develop the deep neural networks consisting of multiple processing layers to extract multiple levels of data features. With deep model structure, raw data are transformed into low level of data features, which are further used to compute higher level of data representations. Deep learning has turned out to be very helpful to discover the intricate information among the high-dimensional data in many fields, including imagine recognition, natural language processing and gene data analysis [6]–[10].

Typical deep learning methods include convolutional neural network (CNN), deep belief network (DBN) and deep autoencoder network (DAN). CNN, inspired by the biological study on the animal visual cortex, is a feed-forward network with a series of convolutional layers and pooling layers [6]. Using CNN, Lecun *et al.* [11] constructed a deep neural network named LeNet-5 for handwritten character recognition. In order to recognize the human actions in videos, Ji *et al.* [12] proposed a 3D CNN by extracting the features from both spatial and the temporal dimensions. DBN, consisting of multiple restricted Boltzmann machines (RBMs), utilizes a layer-by-layer greedy learning strategy to initialize the network parameters and then performs the parameter fine-tuning by combing the desired outputs [13]. Lim *et al.* [14] developed a fingerprint recognition system based on the DBN and validated its effectiveness on the international competition dataset LivNet2013. Abdel-Zaher and Eldeib [15] applied the DBN to construct a computer-aid medical diagnosis system for breast cancer classification. DAN constructs the network structure by stacking multiple layers of autoencoders [8]. Xiong and Zuo [16] used the DAN to recognize the geochemical anomalies. For handling the video-based human pose recovery problem, Hong *et al.* [17] designed a multimodal deep autoencoder and demonstrated its effectiveness with experiments. In the natural language processing field, Zhou *et al.* [18] applied the stacked autoencoders for cross-lingual sentiment classification. It can be seen that deep neural networks have been applied to deal with different problems, and some more application examples of deep learning can be found in [19]–[21].

This paper considers how to apply deep learning to improve the current industrial process monitoring methods. In the data-driven industrial process monitoring field, the commonly used feature extraction tools are the PCA and its nonlinear version known as kernel PCA (KPCA) [22], [23], which represent the current state-of-the-arts in extracting linear and nonlinear features, respectively, and they have been successfully applied to industrial process monitoring and fault detection [4], [24]–[26]. However, PCA and KPCA only compute one single layer of linear or nonlinear features for process monitoring, and they belong to the so-called shallow machine learning techniques. How to integrate the deep learning mechanism with the basic PCA and KPCA components is extremely valuable and may enhance industrial process monitoring performance .

Specifically, motivated by the success of deep learning strategy in many fields, we propose a deep learning based nonlinear PCA model, referred to as deep PCA (DePCA), for industrial process monitoring and fault detection. In the proposed method, a statistical monitoring model is designed with multiple layers of feature extractions so that the hidden

X. Deng and X. Tian (dengxg2002@gmail.com, tianxm@upc.edu.cn) are with College of Information and Control Engineering, China University of Petroleum, Qingdao 266580, China

S. Chen and C.J. Harris (sqc@ecs.soton.ac.uk, cjh@ecs.soton.ac.uk) are with Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, UK. S. Chen is also with King Abdulaziz University, Jeddah 21589, Saudi Arabia.

process information among the process data can be sufficiently exploited. Furthermore, the monitoring results at different layers are fused to indicate the process operating status. Hence, our contribution is two-fold. Firstly, we present a multi-layer principal component (PC) feature extraction framework, which uses PCA and KPCA as the basic feature extraction tools to capture linear and nonlinear PCs hierarchically. To our best knowledge, we are the first to combine PCA and KPCA with deep learning idea in industrial process monitoring. Secondly, we propose a Bayesian inference based multi-layer information fusion. More specifically, with the use of DePCA monitoring model, multi-layer monitoring statistics are obtained based on the different layers of features. Bayesian inference can be used to transform the monitoring statistics at each layer into the fault probabilities, which are weighted to construct two probability-based overall statistics for indicating the process operating status.

The remainder of this paper is organized as follows. Starting with a brief review of PCA and KPCA as the motivator, Section II details the proposed DePCA method for industrial process monitoring application, including the DePCA model structure, the Bayesian inference based monitoring statistics construction, and the process monitoring procedure based on DePCA. Section III provides the case study on the benchmark Tennessee Eastman process to verify the proposed method. We draw the conclusions in Section IV.

II. The proposed DePCA method

A. Motivation

PCA is a well-known machine learning algorithm widely applied to data-based industrial process monitoring. The PCA statistical modeling procedure is depicted in Fig. 1, where the original data $\boldsymbol{X}$ are linearly mapped onto the linear features $\boldsymbol{Y}_L$, and two statistics, T^2 and Q, are used to monitor the changes of the features [22]. The procedure of linear feature extraction and the construction of T^2 and Q statistics can readily be found in the literature, e.g., [22], and they will not be repeated here.

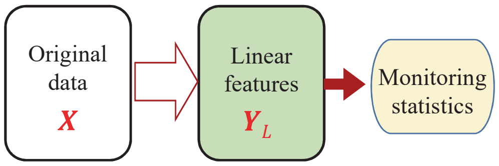

Fig. 1: PCA statistical modelling diagram.

By contrast, KPCA involves kernel mapping in order to extract nonlinear features. As shown in Fig. 2, the original data $\boldsymbol{X}$ are firstly nonlinearly transformed into the kernel matrix $\boldsymbol{K}$ and then the kernel PCs (KPCs) are obtained as the nonlinear features $\boldsymbol{Y}_N$ for process monitoring [23], [25]. Again, the detailed KPC extraction procedure and the construction of the corresponding T^2 and Q statistics based on the extracted nonlinear features can readily be found in the literature, and therefore they will not be repeated here.

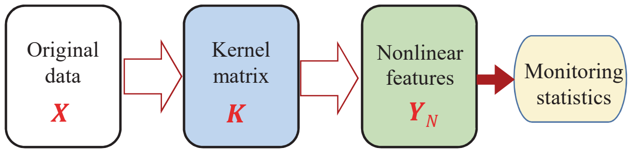

Fig. 2: KPCA statistical modelling diagram.

Comparing Fig. 2 with Fig. 1, it is obvious that the KPCA model has a deeper structure than the PCA. This explains why KPCA performs better than PCA in many industrial process monitoring applications. However, both the PCA and KPCA can be regarded as 'shallow' learning as they involve only one layer of linear and nonlinear features, respectively. Motivated by deep learning theory, we propose a DePCA model, which improves the present PCA and KPCA methods by extracting multiple levels of features from the data.

B. DePCA model structure

The proposed DePCA model structure is shown in Fig. 3, which consists of the $L \geq 2$ feature layers constructed hierarchically. The first feature layer extracts the linear features, while the lth feature layer mines the nonlinear features layer-wise, where $2 \leq l \leq L$. Furthermore, the two monitoring statistics are constructed based on each layer of features. We now detail this DePCA model.

Given the training data matrix $\boldsymbol{X} \in \mathbb{R}^{n\times m}$ with the n samples of m variables, PCA is applied at the first feature layer to seek a projection vector $\boldsymbol{f}^{1)} \in \mathbb{R}^m$ so that the linear transformation $\boldsymbol{y}^{1)} = \boldsymbol{X}\boldsymbol{f}^{1)}$ represents as much information of $\boldsymbol{X}$ as possible, where $^{1)}$ denotes the first-layer feature extraction. Specifically, the optimization to obtain the projection vector $\boldsymbol{f}^{1)}$ is formulated as

$$\begin{aligned} &\max_{\boldsymbol{f}^{1)}} J(\boldsymbol{f}^{1)}) = \max_{\boldsymbol{f}^{1)}} \tfrac{1}{n-1}\left(\boldsymbol{f}^{1)}\right)^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{f}^{1)}, \\ &\text{s.t. } \left(\boldsymbol{f}^{1)}\right)^{\mathrm{T}}\boldsymbol{f}^{1)} = 1. \end{aligned} \tag{1}$$

Solving the optimization problem (1) results in the following eigenvalue problem

$$\frac{1}{n-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{f}^{1)} = \lambda^{1)}\boldsymbol{f}^{1)}, \tag{2}$$

whose solutions are the m eigenvectors $\boldsymbol{f}_j^{1)}$ for $1 \leq j \leq m$ corresponding to the eigenvalues $\lambda_1^{1)} \geq \lambda_2^{1)} \geq \cdots \geq \lambda_m^{1)}$. The first-layer features are computed as $\boldsymbol{y}_j^{1)} = \boldsymbol{X}\boldsymbol{f}_j^{1)}$ for $1 \leq j \leq m$, which yields the first-layer feature matrix $\boldsymbol{Y}^{1)} = \left[\boldsymbol{y}_1^{1)}\ \boldsymbol{y}_2^{1)} \cdots \boldsymbol{y}_m^{1)}\right] \in \mathbb{R}^{n\times m}$.

At the lth feature layer, where $2 \leq l \leq L$, KPCA optimization is performed on the input feature matrix $\boldsymbol{Y}^{l-1)}$ to extract the KPCs $\boldsymbol{Y}^{l)}$ as the lth-layer nonlinear features. Specifically, the input data matrix $\boldsymbol{Y}^{l-1)}$ is implicitly mapped by a nonlinear mapping $\psi(\bullet)$ onto the new high-dimensional space $\mathcal{F}$, and the result is denoted as $\psi\left(\boldsymbol{Y}^{l-1)}\right) \in \mathbb{R}^n \times \mathcal{F}$. Then linear PCA is performed on the matrix $\psi\left(\boldsymbol{Y}^{l-1)}\right)$, and

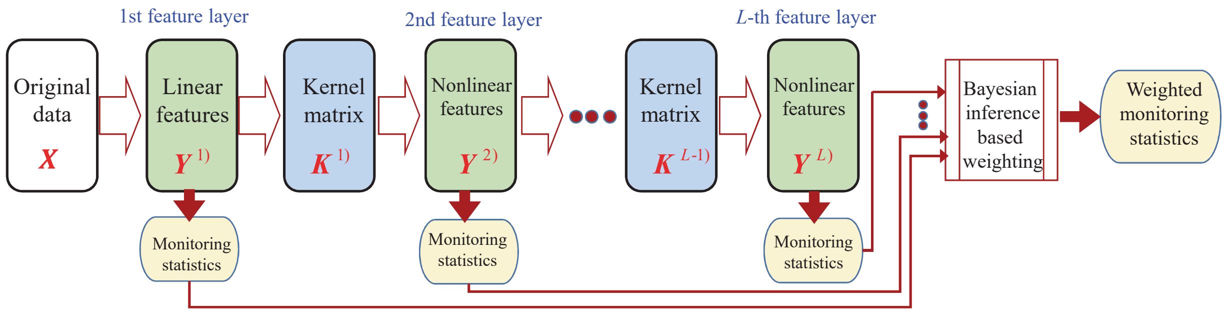

Fig. 3: DePCA statistical modelling diagram.

this yields the following optimization task

$$\begin{aligned}\max_{\boldsymbol{f}^{l)}} J(\boldsymbol{f}^{l)}) &= \max_{\boldsymbol{f}^{l)}} \frac{(\boldsymbol{f}^{l)})^{\mathrm{T}}(\psi(\boldsymbol{Y}^{l-1)}))^{\mathrm{T}}\psi(\boldsymbol{Y}^{l-1)})\boldsymbol{f}^{l)}}{n-1}, \\ \text{s.t. } (\boldsymbol{f}^{l)})^{\mathrm{T}}\boldsymbol{f}^{l)} &= 1,\end{aligned} \tag{3}$$

where $\boldsymbol{f}^{l)} \in \mathcal{F}$ is the PCA projection vector at the lth feature layer. The projection vector $\boldsymbol{f}^{l)}$ can be expressed as the linear combination of $\psi(\boldsymbol{Y}^{l-1)})$ [23], [24]

$$\boldsymbol{f}^{l)} = (\psi(\boldsymbol{Y}^{l-1)}))^{\mathrm{T}}\boldsymbol{\beta}^{l)}, \tag{4}$$

where $\boldsymbol{\beta}^{l)} \in \mathbb{R}^n$ is the coefficient vector. The optimization (3) can thus be reformulated to

$$\begin{aligned}\max_{\boldsymbol{f}^{l)}} J(\boldsymbol{f}^{l)}) = \max_{\boldsymbol{f}^{l)}} \tfrac{1}{n-1}&\Big((\boldsymbol{\beta}^{l)})^{\mathrm{T}}\psi(\boldsymbol{Y}^{l-1)})(\psi(\boldsymbol{Y}^{l-1)}))^{\mathrm{T}} \\ &\times\psi(\boldsymbol{Y}^{l-1)})(\psi(\boldsymbol{Y}^{l-1)}))^{\mathrm{T}}\boldsymbol{\beta}^{l)}\Big), \\ \text{s.t. } (\boldsymbol{\beta}^{l)})^{\mathrm{T}}\psi(\boldsymbol{Y}^{l-1)})&(\psi(\boldsymbol{Y}^{l-1)}))^{\mathrm{T}}\boldsymbol{\beta}^{l)} = 1.\end{aligned} \tag{5}$$

Kernel trick is usually applied to avoid the explicitly defining $\psi(\bullet)$. Denote $\overline{\boldsymbol{y}}_i^{l-1)}$ as the ith sample of $\boldsymbol{Y}^{l-1)}$, i.e., the transpose of the ith row of $\boldsymbol{Y}^{l-1)}$. Then we have the kernel matrix $\boldsymbol{K}^{l-1)} = \psi(\boldsymbol{Y}^{l-1)})(\psi(\boldsymbol{Y}^{l-1)})))^{\mathrm{T}} \in \mathbb{R}^{n\times n}$ with the (i,j)th element $[\boldsymbol{K}^{l-1)}]_{i,j} = (\psi(\overline{\boldsymbol{y}}_i^{l-1)}))^{\mathrm{T}}\psi(\overline{\boldsymbol{y}}_j^{l-1)}) = \ker(\overline{\boldsymbol{y}}_i^{l-1)},\overline{\boldsymbol{y}}_j^{l-1)})$. Here ker$(\bullet,\bullet)$ is the chosen kernel function. Commonly used kernel functions include Gaussian kernel and polynomial kernel [23], [25]. With this kernel trick, the optimization (5) becomes

$$\begin{aligned}\max_{\boldsymbol{f}^{l)}} J(\boldsymbol{f}^{l)}) &= \max_{\boldsymbol{f}^{l)}} \tfrac{1}{n-1}(\boldsymbol{\beta}^{l)})^{\mathrm{T}}\boldsymbol{K}^{l-1)}\boldsymbol{K}^{l-1)}\boldsymbol{\beta}^{l)}, \\ \text{s.t. } (\boldsymbol{\beta}^{l)})^{\mathrm{T}}\boldsymbol{K}^{l-1)}\boldsymbol{\beta}^{l)} &= 1,\end{aligned} \tag{6}$$

whose solutions can be computed by the following eigenvalue decomposition

$$\boldsymbol{K}^{l-1)}\boldsymbol{\beta}^{l)} = (n-1)\lambda^{l)}\boldsymbol{\beta}^{l)}. \tag{7}$$

Solving (7) results in the $n^{l)}$ non-zero eigenvalues $\lambda_1^{l)} \geq \lambda_2^{l)} \geq \cdots \geq \lambda_{n^{l)}}^{l)}$ with the corresponding eigenvectors $\boldsymbol{\beta}_j^{l)}$ for $1 \leq j \leq n^{l)}$, where $n^{l)} \leq n$. Then the lth-layer features are obtained as $\boldsymbol{Y}^{l)} = [\boldsymbol{y}_1^{l)}\ \boldsymbol{y}_2^{l)} \cdots \boldsymbol{y}_{n^{l)}}^{l)}] \in \mathbb{R}^{n\times n^{l)}}$, with $\boldsymbol{y}_j^{l)} = \boldsymbol{K}^{l-1)}\boldsymbol{\beta}_j^{l)}$ for $1 \leq j \leq n^{l)}$.

For a given testing vector $\boldsymbol{x}_t$, its first-layer features $\boldsymbol{y}_t^{1)} = \left[y_{t,1}^{1)}\ y_{t,2}^{1)} \cdots y_{t,m}^{1)}\right]^{\mathrm{T}}$ are calculated according to

$$y_{t,j}^{1)} = \boldsymbol{x}_t^{\mathrm{T}}\boldsymbol{f}_j^{1)},\ 1 \leq j \leq m, \tag{8}$$

while its lth-layer features for $2 \leq l \leq L$, $\boldsymbol{y}_t^{l)} = \left[y_{t,1}^{l)}\ y_{t,2}^{l)} \cdots y_{t,n^{l)}}^{l)}\right]^{\mathrm{T}}$, are computed according to

$$y_{t,j}^{l)} = (\psi(\boldsymbol{y}_t^{l-1)}))^{\mathrm{T}}(\psi(\boldsymbol{Y}^{l-1)}))^{\mathrm{T}}\boldsymbol{\beta}_j^{l)} = (\boldsymbol{k}_t^{l-1)})^{\mathrm{T}}\boldsymbol{\beta}_j^{l)}, \tag{9}$$

for $1 \leq j \leq n^{l)}$, where $\boldsymbol{k}_t^{l-1)} = \psi(\boldsymbol{Y}^{l-1)})\psi(\boldsymbol{y}_t^{l-1)}) \in \mathbb{R}^n$ is the kernel vector corresponding to $\boldsymbol{y}_t^{l-1)}$.

For each layer, two monitoring statistics $T^{2,l)}$ and $Q^{l)}$ are applied to monitor the process changes. Specifically, at the lth layer, the monitoring statistics are constructed as

$$T^{2,l)} = (\boldsymbol{y}_{tk_l}^{l)})^{\mathrm{T}}\boldsymbol{\Lambda}_l^{-1}\boldsymbol{y}_{tk_l}^{l)}, \tag{10}$$

$$Q^{l)} = (\boldsymbol{y}_t^{l)})^{\mathrm{T}}\boldsymbol{y}_t^{l)} - (\boldsymbol{y}_{tk_l}^{l)})^{\mathrm{T}}\boldsymbol{y}_{tk_l}^{l)}, \tag{11}$$

where $\boldsymbol{y}_{tk_l}^{l)} = \left[y_{t,1}^{l)}\ y_{t,2}^{l)} \cdots y_{t,k_l}^{l)}\right]^{\mathrm{T}}$ are the lth-layer PCs corresponding to the first k_l eigenvalues $\lambda_j^{l)}$ for $1 \leq j \leq k_l$, and $\boldsymbol{\Lambda}_l \in \mathbb{R}^{k_l\times k_l}$ is the diagonal matrix with $\lambda_j^{l)}$ for $1 \leq j \leq k_l$ as its diagonal entries. The value of k_l is determined by the average eigenvalue method, which ensures that the first k_l eigenvalues chosen are greater than the average value of all the eigenvalues $\lambda_j^{l)}$, $1 \leq j \leq n^{l)}$ [24].

For the normal operation data, $T^{2,l)}$ and $Q^{l)}$ generally obey some unknown distributions, which however can be estimated by a kernel density estimation (KDE) method [25], [27], [28]. Based on the distributions estimated by KDE, the confidence limits $T_{lim}^{2,l)}$ and $Q_{lim}^{l)}$ can be determined with the given significance level η. In this paper, η is set to 5%. Then $T_{lim}^{2,l)}$ and $Q_{lim}^{l)}$ represent the 95% confidence limits.

C. Bayesian inference based monitoring statistics integration

Based on the DePCA model given in Section II-B, we obtain the monitoring statistics $T^{2,l)}$ and $Q^{l)}$ as well as their confidence limits $T_{lim}^{2,l)}$ and $Q_{lim}^{l)}$, for $1 \leq l \leq L$. In order to construct an overall process status indicator, the monitoring statistics of all the layers are fused together by the Bayesian inference strategy [29]. With this strategy, each monitoring statistic is firstly converted into a posterior fault probability by Bayesian inference. Then all the posterior fault probabilities are weighted to form two probability-based overall monitoring statistics, as illustrated in Fig. 3.

Given the testing vector $\boldsymbol{x}_t$, the posterior fault probabilities related to the statistics $T^{2,l)}$ and $Q^{l)}$ are denoted by $P_{T^2}^{l)}(F|\boldsymbol{x}_t)$

and $P_Q^{l)}(F|\boldsymbol{x}_t)$, respectively, where the symbol F means in fault condition. For notational convenience, we use the symbol S to represent either the T^2 statistic or the Q statistic. The posterior fault probabilities $P_S^{l)}(F|\boldsymbol{x}_t)$ are computed by

$$P_S^{l)}(F|\boldsymbol{x}_t) = \frac{P_S^{l)}(\boldsymbol{x}_t|F)P_S^{l)}(F)}{P_S^{l)}(\boldsymbol{x}_t)}, 1 \leq l \leq L, \tag{12}$$

where $P_S^{l)}(F)$ is the prior fault probability equivalent to the significance level η and $P_S^{l)}(\boldsymbol{x}_t|F)$ is the occurrence probability of $\boldsymbol{x}_t$ given the fault condition, defined by

$$P_S^{l)}(\boldsymbol{x}_t|F) = \exp\big(-\gamma S_{lim}^{l)}/S^{l)}\big), \tag{13}$$

in which γ is the tuning parameter designed to decrease the sensitivity to the outlier data. In this paper, γ is empirically chosen to be 0.2. The occurrence probability of $\boldsymbol{x}_t$ in (12), $P_S^{l)}(\boldsymbol{x}_t)$, can be obtained according to

$$P_S^{l)}(\boldsymbol{x}_t) = P_S^{l)}(\boldsymbol{x}_t|F)P_S^{l)}(F) + P_S^{l)}(\boldsymbol{x}_t|N)P_S^{l)}(N), \tag{14}$$

where N indicates the normal operation, and $P_S^{l)}(N)$ is the prior normal-operation probability with its value as the confidence level $1-\eta$, while $P_S^{l)}(\boldsymbol{x}_t|N)$ is the occurrence probabiltiy of $\boldsymbol{x}_t$ under the normal condition, calculated by

$$P_S^{l)}(\boldsymbol{x}_t|N) = \exp\big(-\gamma S^{l)}/S_{lim}^{l)}\big). \tag{15}$$

Based on the posterior fault probabilities $P_{T^2}^{l)}(F|\boldsymbol{x}_t)$ and $P_Q^{l)}(F|\boldsymbol{x}_t)$ of all the layers, two overall monitoring statistics WT^2 and WQ are constructed by a weighting approach as

$$WT^2 = \sum\nolimits_{l=1}^{L} w_{T^2}^{l)} P_{T^2}^{l)}(F|\boldsymbol{x}_t), \tag{16}$$

$$WQ = \sum\nolimits_{l=1}^{L} w_{Q}^{l)} P_{Q}^{l)}(F|\boldsymbol{x}_t), \tag{17}$$

where $w_{T^2}^{l)}$ and $w_Q^{l)}$ are the weightings, which are calculated based on the conditional fault probabilities as

$$w_{T^2}^{l)} = P_{T^2}^{l)}(\boldsymbol{x}_t|F) \Big/ \sum\nolimits_{i=1}^{L} P_{T^2}^{i)}(\boldsymbol{x}_t|F), \tag{18}$$

$$w_{Q}^{l)} = P_{Q}^{l)}(\boldsymbol{x}_t|F) \Big/ \sum\nolimits_{i=1}^{L} P_{Q}^{i)}(\boldsymbol{x}_t|F). \tag{19}$$

When $WT^2 \leq \eta$ and $WQ \leq \eta$, the process is under the normal operation status. Otherwise, a fault is detected.

D. Fault detection procedure based on DePCA

The DePCA based fault detection procedure includes two stages: model training stage and online monitoring stage. In the first stage, we collect the normal operation data from the process and build the statistical model by the DePCA method. Then in the second stage, the real-time process operating data are projected onto the DePCA model to compute the multilayer features, which are used to construct the monitoring statistics WT^2 and WQ for fault detection.

Model training stage:

1) Collect the process dataset $\boldsymbol{X}$ under the normal operation condition, and divide $\boldsymbol{X}$ into two subsets: model training dataset $\boldsymbol{X}_{tr}$ and model validating dataset $\boldsymbol{X}_{va}$.
2) Normalize all the variables of the dataset $\boldsymbol{X}_{tr}$ to zero mean and unit variance, and perform the DePCA modeling.
3) Normalize the validating dataset $\boldsymbol{X}_{va}$ with the means and variances of the original $\boldsymbol{X}_{tr}$, and compute its projection onto the DePCA model.
4) Obtain the monitoring statistics $T^{2,l)}$ and $Q^{l)}$ as well as their confidence limits $T_{lim}^{2,l)}$ and $Q_{lim}^{l)}$ by the KDE method.

Online monitoring stage:

1) Collect the real-time data sample $\boldsymbol{x}_t$ for online monitoring.
2) Scale $\boldsymbol{x}_t$ with the means and variances of the normal training dataset, and compute its multiple-layer features using the trained DePCA model.
3) Based on the layer-wise features of $\boldsymbol{x}_t$, obtain the monitoring statistics of all the layers and then calculate the weighted overall monitoring statistics WT^2 and WQ.
4) Compare the monitoring statistics WT^2 and WQ with the significance level η to determine if a fault occurs.

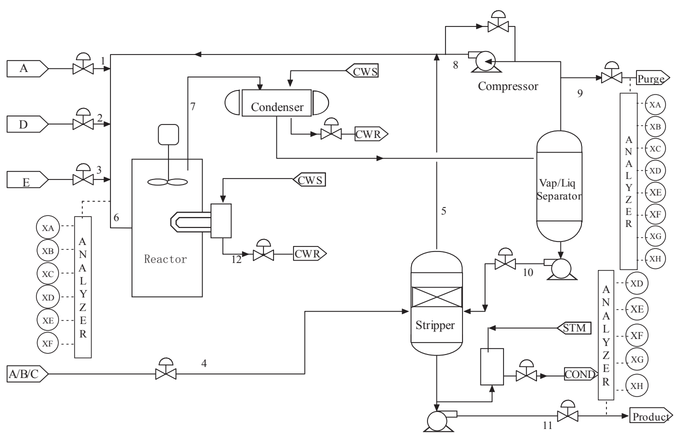

Fig. 4: The Tennessee Eastman process flowchart

III. Case study

The Tennessee Eastman (TE) process is a benchmark process widely adopted to evaluate the process monitoring and fault diagnosis methods [25], [30]–[32]. This process simulates a real chemical process, whose flowchart is depicted in Fig. 4. The TE process involves five major units: reactor, condenser, compressor, stripper and separator. The monitored

TABLE I: List of the 10 TE process faults used in this paper.

Fault label	Description	Fault code	Type
F1	A/C feed ratio(stream 4)	IDV1	Step
F2	Reactor cooling water inlet temperature	IDV4	Step
F3	Condenser cooling water inlet temperature	IDV5	Step
F4	C feed temperature (stream 4)	IDV10	Random
F5	Reactor cooling water inlet temperature	IDV11	Random
F6	Unknown fault	IDV16	Unknown
F7	Unknown fault	IDV17	Unknown
F8	Unknown fault	IDV19	Unknown
F9	Unknown fault	IDV20	Unknown
F10	Stream 4 valve	IDV21	Sticking

33 variables consist of 22 process measurement variables and 11 manipulated variables. The process simulator provides the normal operation case and different fault cases. The 10 faults listed in Table I are applied to validate the proposed DePCA method. The normal and fault datasets can be downloaded from http://web.mit.edu/braatzgroup/links.html. It is noted that the details of faults F6, F7, F8 and F9 are not open to the public but their fault datasets are available. A total of 1460 normal operation samples are collected and they are divided into two subsets. One subset with 500 samples is used as the training dataset and the another subset with 960 samples is used as the validating dataset. Each fault dataset contains 960 samples, and the fault is introduced after the 160th sample.

The proposed DePCA method is applied to monitor the TE process in comparison with the traditional PCA and KPCA based methods. In the monitoring charts, the dashed line represents the 95% confidence limit while the solid curve indicates the monitoring statistic. Two performance metrics, fault detection rate and fault detection time, are used to evaluate the monitoring results. The fault detection rate is defined as the percentage of the samples exceeding the confidence limits over all the faulty samples, while the fault detection time is defined as the sample index of the alarming sample from which the successive 6 samples all exceed the confidence limit.

How 'deep' the DePCA model is specified by the number of layers L. The value of L has significant impact on the achievable performance, and an appropriate L is obviously problem dependent. In the present case study, we empirically set the number of layers to $L = 2$, and the resulting DePCA involves one layer of linear features and one layer of nonlinear features. This choice of DePCA is sufficient to achieve better fault detection performance over the existing state-of-the-art KPCA, as will be shown in the following results. On the other hand, the online computational complexity of this DePCA is very close to that of the KPCA, since the additional complexity involved in extracting linear features is negligible. The kernel function of KPCA and DePCA is selected as the Gaussian kernel with the kernel width set to $\delta = 3300$ empirically. The number of the retained linear or nonlinear PCs is determined by the average eigenvalue method.

We first consider fault F4, which involves the random variations of stream 4 feed temperature. The monitoring charts of the PCA, KPCA and DePCA methods are shown in Figs. 5 to 7, respectively. Observe from Fig. 5 that the PCA T^2 statistic detects this fault at the 218th while its Q statistic indicates the fault at the 208th sample. From Fig. 6, it can be seen that the KPCA T^2 and Q statistics alarm this fault at

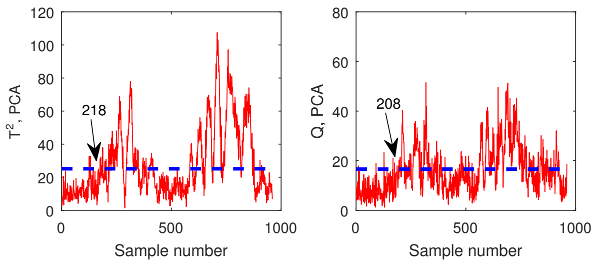

Fig. 5: PCA monitoring charts for fault F4.

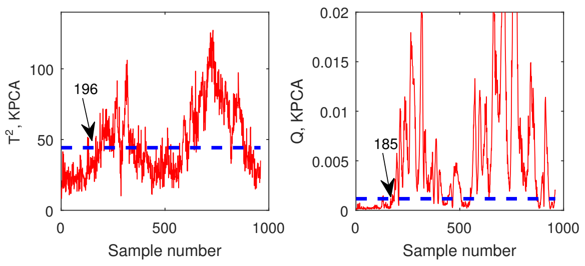

Fig. 6: KPCA monitoring charts for fault F4.

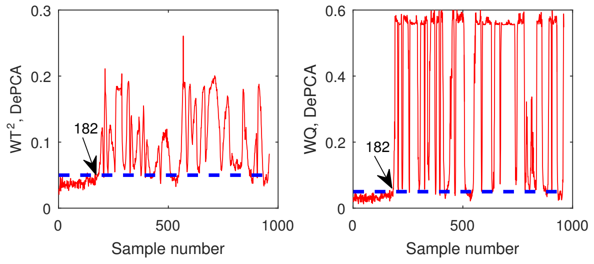

Fig. 7: DePCA monitoring charts for fault F4.

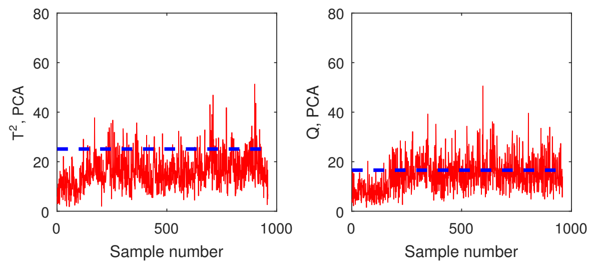

Fig. 8: PCA monitoring charts for fault F8.

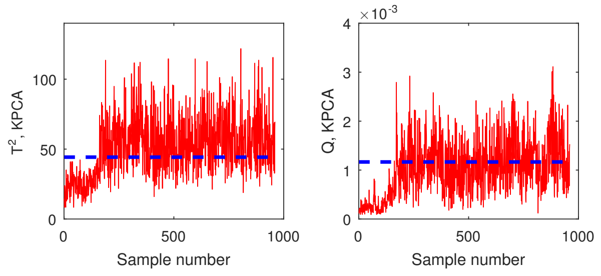

Fig. 9: KPCA monitoring charts for fault F8.

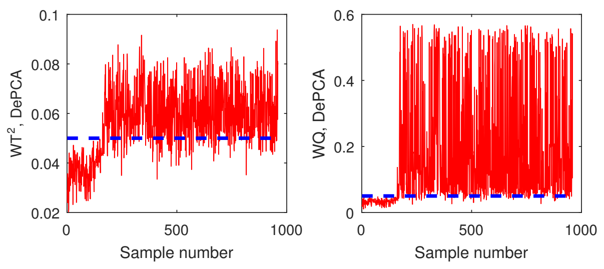

Fig. 10: DePCA monitoring charts for fault F8.

the 196th sample and 185th sample, respectively, which are earlier than those of the PCA method. The DePCA results of Fig. 7 show that the WT^2 and WQ statistics both indicate the fault at the 182th sample. Clearly, the deep learning based DePCA reduces the fault detection delay for fault F4.

The monitoring results of the three methods for fault F8 are shown in Figs. 8 to 10, respectively. It can be seen from Fig. 8 that the PCA T^2 statistic cannot detect this fault effectively, while its Q statistic alarms the fault at the 290-th sample. The fault detection rates of the PCA T^2 and Q statistics are 11.9% and 38.5%, respectively. As shown in Fig. 9, the KPCA T^2 and Q statistics achieve 68.5% and 48.5% of fault detection rate, respectively. Additionally, the KPCA T^2 statistic detects this fault at the 223th sample while its Q statistic gives the alarm signal at the 300th sample. By contrast, the results of Fig. 10 show that the DePCA can detect this fault more quickly than the KPCA, whose WT^2 and WQ statistics indicate the fault at the 171th and 170th sample, respectively. The corresponding fault detection rates are improved to 83.8% and 90.3%, respectively. Clearly, the DePCA considerably outperforms the KPCA in terms of detecting fault F8.

Tables II and III summarize the fault detection rates and fault detection times, respectively, obtained by the PCA, KPCA and DePCA methods for all the 10 TE process faults tested. From Table II, it is observed that the average fault detection rates obtained by the DePCA's WT^2 and WQ statistics are 79.4% and 90.1%, respectively, which are higher than the average fault detection rates of the PCA and KPCA. In Table III, the notation '–' represents fault detection failure.

TABLE II: Fault detection rates (%) of the tested TE process faults obtained by PCA, KPCA and DePCA.

Fault	PCA		KPCA		DePCA	
Label	T^2	Q	T^2	Q	WT^2	WQ
F1	99.3	100	100	99.6	99.9	100
F2	25.8	100	100	83.5	98.6	100
F3	29.5	29.4	29.1	81.0	29.6	100
F4	50.4	45.3	56.1	81.0	86.5	88.8
F5	46.3	79.3	84.1	66.6	71.6	81.8
F6	35.9	46.3	38.5	85.5	89.3	91.6
F7	82.0	95.5	90.6	90.3	94.9	96.6
F8	11.9	38.5	68.5	48.5	83.8	90.3
F9	48.0	58.5	73.4	65.4	88.4	90.8
F10	41.0	55.4	59.5	42.8	51.4	61.3
Average	47.0	64.8	70.0	74.4	79.4	90.1

TABLE III: Fault detection times (sample number) of the tested TE process faults obtained by PCA, KPCA and DePCA.

Fault	PCA		KPCA		DePCA	
Label	T^2	Q	T^2	Q	WT^2	WQ
F1	167	161	161	164	162	161
F2	463	161	161	163	163	161
F3	167	161	161	161	161	161
F4	218	208	196	185	182	182
F5	276	166	166	171	166	166
F6	191	177	350	171	170	167
F7	187	182	182	184	182	182
F8	–	290	223	300	171	170
F9	239	245	239	238	230	230
F10	669	426	416	637	571	416

For faults F1, F2, F3, F5 and F7, the DePCA achieves slight performance enhancements over the KPCA method, but in the cases of faults F4, F6, F8, F9 and F10, the DePCA clearly outperforms the KPCA.

TABLE IV: Average false alarming rates (%) of the tested TE process datasets obtained by PCA, KPCA and DePCA.

Method	PCA		KPCA		DePCA	
Statistics	T^2	Q	T^2	Q	WT^2	WQ
FAR	3.3	4.7	3.9	3.1	2.1	2.2

The false alarming rate (FAR) on the normal operating samples is also an important performance metric for industrial process fault detection. Here we define the FAR as the percentage of the samples exceeding the confidence limit under the normal operation condition. Each of the 10 TE fault datasets contains 160 normal operating samples. The average FARs of the three methods calculated on a total of 1600 normal operating samples are given in Table IV. As 95% confidence limit is used as the detection threshold, up to 5% of the normal operating samples may exceed the confidence limit statistically. From Table IV, it can be seen that all the FARs are lower than 5%. Moreover, the DePCA achieves lower FARs than the PCA and KPCA.

IV. Conclusions

In this paper, inspired by deep learning, a novel DePCA framework has been proposed for industrial process monitoring and fault detection. Different to the traditional PCA and KPCA methods which only monitor one layer of linear or nonlinear features, the proposed method integrates multiple PCA and KPCA components to extract the multiple layer-wise linear and nonlinear data features. A Bayesian inference strategy has been proposed to transform the monitoring statistics of all the feature layers into the posterior fault probabilities. Two comprehensive monitoring statistics have been designed by a weighting strategy, which fuses the posterior fault probabilities from different layers. Simulation results on the benchmark Tennessee Eastman process have validated the superior performance of the proposed DePCA method over the existing state-of-the-art KPCA approach.

Acknowledgment

This work was supported by the Natural Science Foundation of Shandong Province, China (Grant No. ZR2014FL016) and the National Natural Science Foundation of China (Grant Nos. 61403418 and 61273160).

References

[1] M. K. K. Leung, A. Delong, B. Alipanahi, and B. J. Frey, "Machine learning in genomic medicine: a review of computational problems and data sets," *Proc. IEEE*, vol. 104, no. 1, pp. 176–197, Jan. 2016.

[2] A. Chowdhury, E. Kautz, B. Yener, and D. Lewis, "Image driven machine learning methods for microstructure recognition," *Computational Materials Science*, vol. 123, pp. 176–187, Oct. 2016.

[3] S. Chen, X. Hong, E. Khalaf, F. E. Alsaadi, and C. J. Harris, "Complex-valued B-spline neural network and its application to iterative frequency-domain decision feedback equalization for Hammerstein communication systems," in *Proc. IJCNN 2016* (Vancouver, Canada), Jul. 25-29, 2016, pp. 24 – 29.

[4] S. Yin, X. Li, H. Gao, and O. Kaynak, “Data-based techniques focused on modern industry: an overview,” *IEEE Trans. Industrial Electronics*, vol. 62, no. 1, pp. 657–667, Jan. 2015.

[5] M. A. Alsheikh, S. Lin, D. Niyato, and H.-P. Tan, “Machine learning in wireless sensor networks: algorithms, strategies, and applications,” *IEEE Communication Surveys & Tutorials*, vol. 16, no. 4, pp. 1996–2018, 2014.

[6] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[7] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew, “Deep learning for visual understanding : a review,” *Neurocomputing*, vol. 187, pp. 27–48, 2016.

[8] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: a review and new perspectives,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[9] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.

[10] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[12] S. Ji, W. Xu, M. Yang, and K. Yu, “3D convolutional neural networks for human action recognition,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221–231, Jan. 2013.

[13] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A fast learning algorithm for deep belief nets,” *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006.

[14] S. Kim, B. Park, B. S. Song, and S. Yang, “Deep belief network based statistical feature learning for fingerprint liveness detection,” *Pattern Recognition Letters*, vol. 77, pp. 58–65, 2016.

[15] A. M. Abdel-zaher and A. M. Eldeib, “Breast cancer classification using deep belief networks,” *Expert Systems With Applications*, vol. 46, pp. 139–144, 2016.

[16] Y. Xiong and R. Zuo, “Recognition of geochemical anomalies using a deep autoencoder network,” *Computers and Geosciences*, vol. 86, pp. 75–82, Jan. 2016.

[17] C. Hong, J. Yu, J. Wan, D. Tao, and M. Wang, “Multimodal deep autoencoder for human pose recovery,” *IEEE Trans. Image Processing*, vol. 24, no. 12, pp. 5659–5670, Dec. 2015.

[18] G. Zhou, Z. Zhu, T. He, and X. T. Hu, “Cross-lingual sentiment classification with stacked autoencoders,” *Knowledge and Information Systems*, vol. 47, no. 1, pp. 27–44, Apr. 2016.

[19] C. Xia, F. Qi, and G. Shi, “Bottom-up visual saliency estimation with deep autoencoder-based sparse reconstruction,” *IEEE Trans. Neural Networks and Learning Systems*, vol. 27, no. 6, pp. 1227–1240, Jun. 2016.

[20] D. L. K. Yamins and J. J. Dicarlo, “Using goal-driven deep learning models to understand sensory cortex,” *Nature Neuroscience*, vol. 19, no. 3, pp. 356–365, Mar. 2016.

[21] S. Gao, Y. Zhang, K. Jia, J. Lu, and Y. Zhang, “Single sample face recognition via learning deep supervised autoencoders,” *IEEE Trans. Information Forensics and Security*, vol. 10, no. 10, pp. 2108–2118, Oct. 2015.

[22] S. J. Qin, “Survey on data-driven industrial process monitoring and diagnosis,” *Annual Reviews in Control*, vol. 36, no. 2, pp. 220–234, Dec. 2012.

[23] B. Schölkopf, A. Smola, and K.-R. Müller, “Nonlinear component analysis as a kernel eigenvalue problem,” *Neural Computation*, vol. 10, no. 5, pp. 1299–1319, Jul. 1998.

[24] J.-M. Lee, C. Yoo, S. W. Choi, P. A. Vanrolleghem, and I.-B. Lee, “Nonlinear process monitoring using kernel principal component analysis,” *Chemical Engineering Science*, vol. 59, no. 1, pp. 223–234, Jan. 2004.

[25] 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 and Intelligent Laboratory Systems*, vol. 127, pp. 195–209, 2013.

[26] C. Zhao and F. Gao, “Fault-relevant principal component analysis (FPCA) method for multivariate statistical modeling and process monitoring,” *Chemometrics and Intelligent Laboratory Systems*, vol. 133, pp. 1–16, 2014.

[27] X. Hong, S. Chen, and V. M. Becerra, “Sparse density estimator with tunable kernels,” *Neurocomputing*, vol. 173, no. 3, pp. 1976–1982, 2016.

[28] S. Chen, X. Hong, and C. J. Harris, “Probability density estimation with tunable kernels using orthogonal forward regression,” *IEEE Trans. Systems, Man and Cybernetics–Part B: Cybernetics*, vol. 40, no. 4, pp. 1101–1114, Aug. 2010.

[29] W. Shao and X. Tian, “Adaptive soft sensor for quality prediction of chemical processes based on selective ensemble of local partial least squares models,” *Chemical Engineering Research and Design*, vol. 95, pp. 113–132, Mar. 2015.

[30] J. J. Downs and E. F. Vogel, “A plant-wide industrial problem process,” *Computers & Chemical Engineering*, vol. 17, no. 3, pp. 245–255, Mar. 1993.

[31] 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, 2016.

[32] B. Chebel-Morello, S. Malinowski, and H. Senoussi, “Feature selection for fault detection systems: application to the Tennessee Eastman process,” *Applied Intelligence*, vol. 44, no. 1, pp. 111–122, Jan. 2016.