Cloud-Edge Collaborative Method for Industrial Process Monitoring Based on Error-Triggered Dictionary Learning

Keke Huang, Zui Tao, Chen Wang, Tianxu Guo, Chunhua Yang, *Fellow, IEEE*, and Weihua Gui

***Abstract*—The development of cloud manufacturing enables data-driven process monitoring methods to reflect the real industrial process states accurately and timely. However, traditional process monitoring methods cannot update learned models once they are deployed to edge devices, which leads to model mismatch when confronted time-varying data. In addition, limited resources on the edge prevent it from deploying complex models. Therefore, this article proposes a novel cloud-edge collaborative process monitoring method. First, historical data of industrial processes are collected to establish a dictionary learning model and train the dictionary and classifier in the cloud. Then, the model is simplified and deployed to the edge. The edge layer monitors the process states, including fault detection and working condition recognition, and determines whether a model mismatch has occurred based on an error-triggered strategy. Both numerical simulation and industrial roasting process results verify the superiority of the proposed method.**

***Index Terms*—Cloud-edge collaboration, dictionary learning, model simplification, model updating, process monitoring.**

I. Introduction

CLOUD manufacturing (CMfg) adopts and extends the concept of cloud computing to combine advanced information technology, manufacturing technology, and Internet of Thing (IoT) technology to form a shared, scalable manufacturing resource pool [1]–[3], which facilitates the transformation of manufacturing modes, technologies, and ecosystems. Industrial process monitoring is a typical application in CMfg. The widespread deployment of sensors and IoT devices in industrial manufacturing sites has spawned an explosion of huge and diverse industrial data. Process monitoring provides real-time monitoring of the states of industrial processes such as working conditions and fault information through data mining, statistical modeling, and machine learning [4].

Nowadays, process monitoring has been widely studied in academia and industry because of its great importance to ensure the efficient and healthy operation of industrial production and to achieve intelligent manufacturing [5]. Among various process monitoring methods, the most widely used methods are the multivariate statistical process monitoring (MSPM) methods. Typical MSPM methods include principal component analysis (PCA) and partial least square [6], [7]. With the development of artificial intelligence, some machine learning methods have been introduced for process monitoring, such as support vector machine (SVM)-based methods [8] and slow feature analysis methods [9].

Dictionary learning is a machine learning method that extracts features from the data and represents them by dictionary atoms. The K-SVD algorithm [10] achieves the training of dictionary that makes the best representation of each training data, which greatly contributes to the development of dictionary learning. Based on the K-SVD algorithm, some extension methods, such as the label consistent (LC-K-SVD) method [11], the discriminative K-SVD method [12], etc., were developed. Recently, dictionary learning methods have been investigated in the community of process monitoring due to their powerful performance in industrial data representation and classification. Lu *et al.* [13] proposed a physics-constrained dictionary learning method to perform diagnosis with the classification of sensor data. Huang *et al.* [14] proposed a kernel dictionary learning method for nonlinear process monitoring and applied it to aluminum electrolysis industrial process. Yang *et al.* [15] proposed a robust dictionary learning method to remove outliers and noise from training data for fault detection and classification of industrial processes.

Manuscript received 7 December 2021; revised 12 February 2022; accepted 20 March 2022. Date of publication 23 March 2022; date of current version 30 September 2022. This work was supported in part by the National Key R&D Program of China under Grant 2019YFB1705300, in part by the National Natural Science Foundation of China, under Grant 62073340, in part by the Major Key Project of Peng Cheng Laboratory (PCL) under Grant PCL2021A09, in part by the Science and Technology Innovation Program of Hunan Province under Grants 2021RC3018 and 2021RC4054, in part by the Innovation Driven Plan in Central South University, China, under Grant 2019CX020, and in part by the 111 Project, China, under Grant B17048. Paper no. TII-21-5447. *(Corresponding author: Chen Wang.)*

Keke Huang is with the School of Automation, Central South University, Changsha 410083, China, and also with the Pengcheng Laboratory, Shenzhen 518055, China (e-mail: huangkeke@csu.edu.cn).

Zui Tao, Chunhua Yang, and Weihua Gui are with the School of Automation, Central South University, Changsha 410083, China (e-mail: taozui21@csu.edu.cn; ych@csu.edu.cn; gwh@csu.edu.cn).

Chen Wang is with the National Engineering Laboratory for Big Data Software, Tsinghua University, Beijing 100084, China (e-mail: wang_chen@tsinghua.edu.cn).

Tianxu Guo is with the BNRist, Tsinghua University, Beijing 100084, China (e-mail: guotianxu2011@qq.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TII.2022.3161640.

Digital Object Identifier 10.1109/TII.2022.3161640

Although the aforementioned methods have been investigated for some industrial cases, how to apply these advanced process monitoring methods to industrial sites is still a problem worth further investigating in practice. Generally, traditional process monitoring methods deploy models in edge devices through edge computing, since the edge layer is closer to industrial manufacturing sites and has high real-time performance [16]. Unfortunately, general edge devices often have limitations in computing and storage resources and cannot handle large-scale data and complex algorithm models. Moreover, the model deployed in the edge devices cannot be updated adaptively. With the development of CMfg, some process monitoring applications were migrated to the cloud layer since it can handle large-scale industrial data [17], [18]. Cloud computing has less limitation of computing and storage resources than the edge, it can update the model based on real-time data. But the cloud-based process monitoring usually confronted with poor real-time performance due to the time-consuming data transmission. More specifically, the monitoring results are difficult to provide real-time guidance to the production because of the communication delay between the cloud and the industrial site.

The cloud-edge collaboration framework realizes the complementary advantages of edge intelligence and cloud computing [19], [20]. The cloud-edge collaboration framework features the migration of complex model training and Big Data management to the cloud, while the edge is responsible for relatively simple real-time inference. This framework greatly frees up the pressure of computing and storage capacity at the edge, which has great application implications for current process monitoring systems. As a result, there are already some works of process monitoring under the cloud-edge coordination framework [21], [22]. In our work, the cloud is responsible for managing industrial data and training and updating the models. The edge devices use the models trained in the cloud to achieve real-time process monitoring. The difference between our work and that mentioned previously is that in our method the collaboration between the cloud and the edge is more frequent and tighter, as the model needs to be continuously updated.

Our could-edge collaboration framework is ideally suited to address practical problems when it confronts complex industrial process monitoring. In real industrial sites, the process is often characterized by time varying due to the variation of process parameters, raw materials. Since the process monitoring model is trained on historical data, it cannot represent real-time data well, which is also known as a model mismatch. Our method can address model mismatch by updating the model using real-time data adaptively. Take the roaster in practice, the temperature fluctuation in the furnace and the frequent abnormal working conditions lead to drastic time-varying data, which makes the problem of model mismatch. In addition, the harsh environment of the roaster makes it unrealistic for the deployment of complex monitoring algorithms on the site. Therefore, as shown in Fig. 1, it is urgent to propose a monitoring method that is not only applicable in the site but also can update the model. From this practical perspective, this article proposes a cloud-edge collaborative method for industrial process monitoring based on error-triggered dictionary learning. Here, the dictionary learning

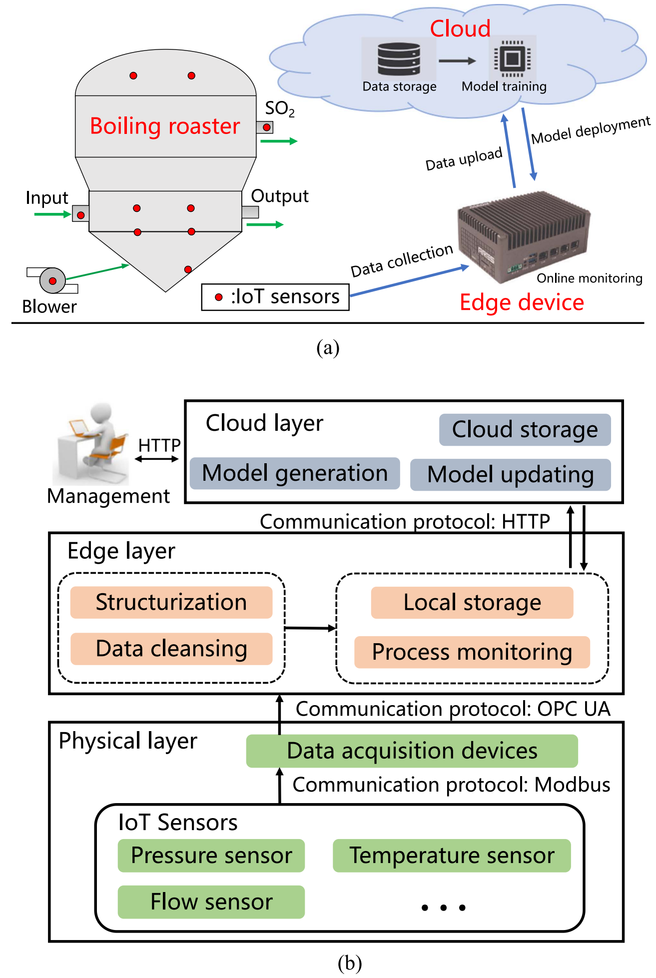

Fig. 1. Cloud-edge collaborative monitoring of boiling roaster. (a) Physical architecture of the cloud-edge collaborative framework. (b) Network architecture of the cloud-edge collaborative framework.

method ensures the accuracy of process monitoring; the cloud-edge collaboration framework provides a solution for online monitoring and model update to address the model mismatch. In detail, a dictionary learning model is built in the cloud to train a dictionary and a classifier. Considering the limited resources of computing and storage on the edge layer, the model is simplified according to the used number of each dictionary atom to obtain the simplified dictionary and classifier. The simplified model is deployed to the edge. As real-time data are captured by the edge, the simplified model is used to monitor the states of the industrial process, including fault detection and working condition recognition. The edge also determines whether a model mismatch occurs based on the calculated label information entropy. If the label information entropy is less than the threshold, it triggers the cloud to update the model to ensure that the model always matches the data. In summary, the main contributions of this article are as follows.

1) A novel cloud-edge collaborative framework combining model update in the cloud and trigger strategy in the edge is proposed to guarantee the accuracy of process monitoring, especially for time-varying cases.

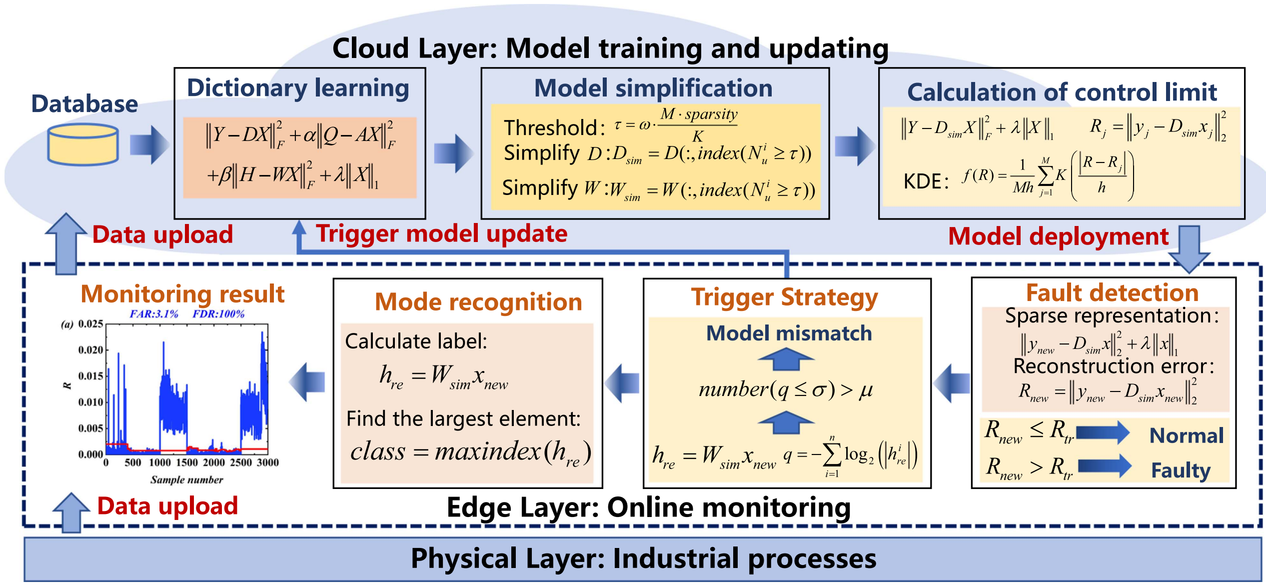

Fig. 2. Cloud-edge collaborative monitoring framework based on the error-triggered dictionary learning method.

2) For high real-time requirements and strict computation and storage limitations of the edge layer, a model simplification method based on the contribution of the dictionary atom is proposed to simplify the dictionary and classifier.
3) An event-triggered strategy based on label error is proposed for solving the model mismatch puzzle. The joint optimization of dictionary and classifier ensures the rationality of the strategy.

The rest of this article is structured as follows. In Section II, the proposed process monitoring method and the framework of cloud-edge collaboration are described in detail. In Section III, the superiority of the proposed method is demonstrated by a numerical simulation and an industrial roasting process. Finally, Section IV concludes this article.

II. Methodology

In this section, we introduce the error-triggered dictionary learning monitoring model and the cloud-edge collaborative monitoring framework in four parts, including cloud-based dictionary learning modeling, model simplification, error-triggered model update strategy, and edge layer online monitoring. The framework structure diagram of the proposed method is shown in Fig. 2.

A. Cloud-Based Dictionary Learning Method

For the cloud-edge collaborative framework, the cloud is mainly responsible for storing large amounts of process data, as well as training and updating models based on the data. Therefore, a suitable data representation model is a prerequisite. Due to the powerful representation ability of dictionary learning, we use it as the baseline model [11]. In addition, industrial processes often operate under different working conditions, the dictionary learning model should be discriminative, namely, the learned dictionary can incorporate the label information to recognize working conditions. In the following, the procedure of data modeling, as well as optimization is described in detail.

First, the historical data from industrial processes are collected and stored in the cloud. Mathematically, the data are denoted as $Y = [y_1, y_2, \ldots, y_M] \in R^{m\times M}$. Here, m and M denotes the dimension and the number of data samples. y_i denotes the data in the i_{th} moment. Since the data come from multiple sensors at the industrial site, it is an m-dimensional vector. The label of the data is denoted as $H = [h_1, h_2, \ldots, h_M] \in R^{n\times M}$. Here, n denotes the total working conditions of historical data. h_i denotes the label vector corresponding to the i_{th} data. If the i_{th} data belong to the j_{th} working condition, $h_i = [0, \cdots 0, 1, 0, \cdots 0]$, of which the j_{th} element of h_i is 1 and the rest elements are 0.

The principle of dictionary learning is to learn a basis set $D = [d_1, d_2, \ldots, d_K] \in R^{m\times K}$ to represent training data, which also known as a dictionary. Here, d_i denotes the i_{th} atom and K denotes the number of atoms. In general, the normalization is performed for each atom, $\|d_i\|_2^2 = 1\, \forall i$. Each data can be represented as a linear combination of a few atoms, i.e., $y = Dx + e$. Here, x denotes the coding coefficient and e denotes the reconstruction error. When the optimal dictionary is built, the error is ignorable. Since the learned dictionary is overcompleted, only a few atoms are used, so x is a sparse vector with only a few nonzero elements. Extending to the entire training set, we obtain $Y = DX$, where $X = [x_1, x_2, \ldots, x_M] \in R^{K\times M}$ is the sparse coding matrix. To train the dictionary and sparse coding matrix, dictionary learning can be transformed into the following optimization problem.

$$\langle D, X\rangle = \arg\min_{D,X} \left\{ \|Y - DX\|_F^2 + \lambda \|X\|_1 \right\} \tag{1}$$

where $\|Y - DX\|_F^2$ denotes the reconstruction error, with $\|A\|_F = \sqrt{\sum_{i,j} a_{ij}^2}$ denotes the Frobenius norm of the matrix A, and $\|X\|_1$ denotes the sparsity constraint on the coding matrix, with $\|A\|_1 = \sum_{i,j} |a_{ij}|$ denotes the L_1 norm of the matrix A. These two terms guarantee that the model has minimal reconstruction error and sparse data representation. Equation (1) is a typical dictionary learning optimization function and is solved using the K-SVD algorithm [10].

However, the dictionary atoms in (1) can represent the training data but cannot learn the category information. We want the atoms to be characterized by discriminative category features. The data of a class is represented using the atoms of this class. Therefore, the following optimization function is developed:

$$\langle D, A, X\rangle = \arg\min_{D,A,X} \times \left\{\|Y - DX\|_F^2 + \alpha \|Q - AX\|_F^2 + \lambda\|X\|_1\right\} \tag{2}$$

where $Q = [q_1, q_2, \ldots, q_M] \in R^{K\times M}$ is called discriminative sparse coding, whose role is to incorporate label information into the learning process. Let q_{ij} be the element of the i_{th} row and j_{th} column of the matrix Q. q_{ij} is equal to 1 only if the label of the i_{th} training data and the j_{th} dictionary atom are the same, otherwise it is 0. $A \in R^{K\times K}$ is a transformation matrix that makes the coding matrix approximate to the discriminative coding matrix Q, thus, promotes the atoms to learn category features.

Then, a linear classifier is used to reconstruct the category labels of the data to make the model can be used for classification, and the following optimization function is obtained:

$$\langle D, A, W, X\rangle = \arg\min_{D,A,W,X} \left\{ \begin{array}{l} \|Y - DX\|_F^2 + \alpha \|Q - AX\|_F^2 \\ +\beta \|H - WX\|_F^2 + \lambda\|X\|_1 \end{array} \right\} \tag{3}$$

where $W \in R^{n\times K}$ denotes a linear classifier that reconstructs the labels by the linear transformation WX, and the data can be classified with the reconstructed labels. The cloud-based dictionary learning baseline model is shown in (3), which has the capability of both data representation and classification and is well suited for cloud-edge collaborative process monitoring in this article.

Before optimization, the initialization of $D^{(0)}$, $A^{(0)}$, and $W^{(0)}$ is considered. We train the dictionary for each class using the K-SVD algorithm, and then, combine the dictionaries of all classes together to obtain the initial dictionary $D^{(0)}$. Initialize $A^{(0)}$ by solving the following optimization problem:

$$A = \arg\min_{A} \left\{\|Q - AX\|_F^2 + \lambda_1 \|A\|_F^2\right\} \tag{4}$$

where the regular term $\|A\|_F^2$ can effectively prevent overfitting. Since the Frobenius norm is convex, the solution formula of $A^{(0)}$ can be obtained by finding the partial derivative of A as

$$A^{(0)} = QX^T(XX^T + \lambda_1 I)^{-1}. \tag{5}$$

Similarly, the solution formula of $W^{(0)}$ can be obtained as

$$W^{(0)} = HX^T(XX^T + \lambda_2 I)^{-1}. \tag{6}$$

After initialization, the model of (3) can be optimized. First, (3) can be rewritten as

$$\min_{D,A,W,X} \left\{ \left\| \begin{pmatrix} Y \\ \sqrt{\alpha}Q \\ \sqrt{\beta}H \end{pmatrix} - \begin{pmatrix} D \\ \sqrt{\alpha}A \\ \sqrt{\beta}W \end{pmatrix} X \right\|_F^2 + \lambda\|X\|_1 \right\}. \tag{7}$$

Let $Y_c = (Y^T, \sqrt{\alpha}Q^T, \sqrt{\beta}H^T)^T$ and $D_c = (D^T, \sqrt{\alpha}A^T, \sqrt{\beta}W^T)^T$. The optimization function of (7) can be simplified as follows:

$$\langle D_c, X\rangle = \arg\min_{D_c,X} \left\{\|Y_c - D_c X\|_F^2 + \lambda\|X\|_1\right\}. \tag{8}$$

The aforementioned equation is a basic dictionary learning optimization function that can be solved using the K-SVD algorithm. After the optimization step, we decompose D_c to get D and W.

B. Model Simplification

Since the model is eventually deployed and used in edge devices, the suitability of the edge devices for the model needs to be taken into account. In general, edge devices have limited computing and storage resources. If the model is too complex, it will inevitably take up too much storage space, and meanwhile, the complex model will also increase the computational complexity, which reduces the real-time performance of monitoring. In addition, since the optimal dictionary size is not known in advance, we tend to set the dictionary as an overcomplete dictionary in the standard dictionary learning setup, i.e., for $D \in R^{m\times K}$, $K \gg m$, which leads to a lot of redundant basis vectors in the dictionary and makes it take too much computation time for data representation. Therefore, it is reasonable and necessary to simplify the dictionary model before deploying it to the edge.

Based on the principle of sparse representation, each training data can be represented as a linear combination of a few atoms. The more an atom contributes to the data representation, the more often it is used. Therefore, after obtaining the sparse representation of the training data, the used number of each atom are counted as $N_u^i, i \in [1, 2, \ldots, K]$. N_u^i is calculated as follows:

$$N_u^i = \left\|x_T^i\right\|_0 \tag{9}$$

where x_T^i denotes the i_{th} row of the coding matrix X. $\|\|_0$ denotes the L_0 norm, which counts the number of nonzero elements in the vector. In dictionary simplification, atoms with high contribution to the data representation are reserved and those with low contribution are deleted. This is achieved by setting a threshold τ for N_u, and τ is calculated as follows:

$$\tau = \omega \cdot \frac{M \cdot \text{sparsity}}{K} \tag{10}$$

where M denotes the number of training samples, sparsity denotes the sparsity in data representation, and K denotes the number of atoms. ω denotes a weight. If N_u^i is less than τ, it means that the i_{th} atom is used relatively infrequently, then the atom and the corresponding column in W can be deleted. Accordingly, the simplified dictionary D_{sim} and classifier W_{sim}

Algorithm 1: Cloud-based dictionary learning modeling and simplification.

Input: Industrial process data Y, label H, discriminative sparse coding Q, number of atoms, coefficient α, β, ω, and sparsity
Initialization: Subdictionaries are trained using the K-SVD algorithm for each class of data and combined to obtain $D^{(0)}$. Use (5) and (6) to calculate $A^{(0)}$ and $W^{(0)}$
Step 1: $Y_c = (Y^T, \sqrt{\alpha}Q^T, \sqrt{\beta}H^T)^T$, $D_c = (D^T, \sqrt{\alpha}A^T, \sqrt{\beta}W^T)^T$;
Step 2: Update D_c using the K-SVD algorithm;
Step 3: Decompose D_c to get D and W;
Step 4: Calculate the used number of each atom in the training data;
Step 5: Use (10) to calculate the threshold τ;
Step 6: Use (11) and (12) to obtain the simplified dictionary D_{sim} and the simplified classifier W_{sim};
Output: D_{sim} and W_{sim}

can be calculated mathematically as follows:

$$D_{\text{sim}} = D(:, \text{index}(N_u^i \geq \tau)) \tag{11}$$

$$W_{\text{sim}} = W(:, \text{index}(N_u^i \geq \tau)) \tag{12}$$

where $\text{index}(N_u^i \geq \tau)$ denotes the index of the atom whose used number is more than τ. The threshold τ can impact the size of the simplified model. If τ is larger, more atoms will be deleted, then D_{sim} and W_{sim} will be smaller, while conversely D_{sim} and W_{sim} will be larger. The size of D_{sim} will impact the effect of edge layer monitoring, so it is necessary to find suitable D_{sim} and W_{sim} by adjusting ω.

In summary, a dictionary learning model is built based on historical data in the cloud to train and simplify a dictionary and a classifier. The simplified model is deployed at the edge for real-time process monitoring. The detailed steps of model training and simplification are summarized in Algorithm 1.

C. Error-Triggered Model Update Strategy

In the actual industrial site, the process data are always characterized by time varying due to factors such as changes in process parameters and raw materials, which causes a deviation between real-time data and historical data and results in models trained on historical data not matching the real-time data. Model mismatch leads to much of the normal data may be falsely alarmed. Therefore, it is necessary to update the model so that it always matches the time-varying data.

However, in the traditional process monitoring methods, the model is fixed once it has been deployed to the edge device and cannot be updated adaptively due to the limitations of computing power. The cloud-edge collaborative framework solves this problem by placing the training and updating of the model in the cloud with powerful computing power, then simply deploying the updated model to the edge, which greatly frees up computing resources of the edge and makes the model updating feasible.

The edge layer should provide the triggering strategy. The two commonly used strategies at present are timed-triggered and event-triggered updating. Timed-triggered updating means that a fixed time or amount of data is given, the model is updated after the system has run for a fixed time or processed a fixed amount of data. Event-triggered updating means that the system checks to see if a model mismatch event has occurred to trigger the model update. In real industrial processes, the moment of model mismatch is often uncertain. Time-triggered strategies usually have difficulty in finding the exact moment of mismatch, which leads to the system being monitored by a mismatched model for a long period of time. The event-triggered strategy, on the other hand, detects whether the model mismatch occurs at each sampling cycle, which is relatively faster to find the moment of model mismatch and trigger an update in time so that the model always matches the data, and is better in practice. In addition, event-triggered strategy can reduce the communication traffic and save computational load on the processors. Thus, the event-triggered updating strategy is introduced in this article.

The dictionary D for data representation and the classifier W for classification are jointly optimized, which promotes the features extracted by the model is jointly for data representation and classification. Therefore, the error of the label given by the classifier is consistent with the ability of the data representation by the dictionary. When the model mismatch causes the data representation to deteriorate, the error of the labels given by the classifier will also deteriorate. Therefore, label error is used to determine whether a model mismatch has occurred. When monitoring online data, the label $h_{\text{re}} = [h_{\text{re}}^1, h_{\text{re}}^2, \ldots, h_{\text{re}}^n]$ can be reconstructed. Then, the label information entropy is calculated as follows:

$$q = -\sum_{i=1}^{n} \log_2 \left(\left| h_{\text{re}}^i \right| \right). \tag{13}$$

We use the one-hot code as the label. Information entropy is often used to determine the quality of the one-hot code labels, especially in some dictionary learning work [23]. For one-hot code, when the reconstructed label is good, with only one element close to 1 and the other elements close to 0, the information entropy will be larger. Conversely, a bad label vector will have a smaller difference between the values of its different elements, and in this case, a smaller information entropy. Therefore, q can be used to determine whether a label passes or fails. Set a threshold σ for q to measure the error of the label, when $q > \sigma$, the label passes, otherwise the label fails. However, a label failure is not necessarily a model mismatch, but can also be a coincidental event. Therefore, an upper limit μ on the number of failed labels is preset. When the number of failed labels is greater than μ, an update of the model in the cloud is triggered.

In general, retraining and incremental learning can be used for model updating. As there is no limitation of computing resources in the cloud, we do not need an incremental update strategy. In this article, real-time data are added to the training set and the model is updated by retraining.

D. Edge Layer Online Monitoring

At the edge layer, we use the reconstruction error of the online data to determine whether a fault has occurred, so the control limit for the reconstruction error is calculated. The first step is to obtain the sparse representation of D_{sim} to the training data as follows:

$$X = \arg\min_{X} \left\{ \|Y - D_{\text{sim}}X\|_F^2 + \lambda \|X\|_1 \right\}. \tag{14}$$

The aforementioned equation can be solved by the orthogonal matching pursuit (OMP) method [24]. After obtaining the sparse representation, the reconstruction errors can be calculated as follows:

$$R_j = \|y_j - D_{\text{sim}}x_j\|_2^2 \tag{15}$$

where y_j, x_j, and R_j denote the j_{th} training data, sparse coding, and reconstruction error. After calculating the reconstruction errors of all training data, the probability density function of the reconstruction errors is estimated by the kernel density estimation (KDE) method [25]. The formula of KDE is as follows:

$$f(R) = \frac{1}{Mh} \sum_{j=1}^{M} K\left(\frac{|R - R_j|}{h}\right) \tag{16}$$

where h denotes the bandwidth and $K(x)$ denotes the kernel function; the Gaussian kernel function is chosen in this article. Once the probability density function $f(R)$ has been obtained, the control limit R_{tr} for the reconstruction error is obtained by setting an appropriate confidence level ε. The calculation formula is as follows:

$$\int_{-\infty}^{R_{\text{tr}}} f(R)dR = \frac{\varepsilon}{2}. \tag{17}$$

When the model is updated, the control limit is also updated.

1) Fault Detection: When the online monitoring data y_{new} is collected by the sensors and sent to the edge device, it can be monitored at the edge. Use D_{sim} to represent it.

$$x_{\text{new}} = \arg\min_{x} \left\{ \|y_{\text{new}} - D_{\text{sim}}x\|_2^2 + \lambda \|x\|_1 \right\}. \tag{18}$$

Similarly, x_{new} is calculated using the OMP method. Afterward, the reconstruction error is calculated as follows:

$$R_{\text{new}} = \|y_{\text{new}} - D_{\text{sim}}x_{\text{new}}\|_2^2. \tag{19}$$

Since the faulty data cannot be represented by the dictionary, the reconstruction error will be large. Therefore R_{new} is compared with R_{tr}. If $R_{\text{new}} \leq R_{\text{tr}}$, the online data are normal. Otherwise, it is faulty.

2) Working Condition Recognition: If the online data are normal, it is possible to recognize working conditions after fault detection. x_{new} has been calculated, and the label vector h_{re} can be reconstructed by the classifier W_{sim}.

$$h_{\text{re}} = W_{\text{sim}}x_{\text{new}}. \tag{20}$$

The index of the largest element in the label vector is the working condition to which the data belongs. If the i_{th} element in the vector h_{re} is the largest, then the data belongs to the i_{th} working condition.

III. Illustrative Examples

In this section, examples of numerical simulation and industrial roasting process are used to verify the superiority of the proposed cloud-edge collaborative process monitoring method. To quantitatively assess the monitoring effectiveness, some indicators are defined as follows, including the false alarm rate (FAR) and the fault detection rate (FDR).

$$\text{FDR} = \frac{\#\{R > R_{\text{th}} | f \neq 0\}}{\#\{f \neq 0\}} \times 100\% \tag{21}$$

$$\text{FAR} = \frac{\#\{R > R_{\text{th}} | f = 0\}}{\#\{f = 0\}} \times 100\% \tag{22}$$

where $\#\{x\}$ denotes the number of samples in the set x, $f \neq 0$ denotes faulty data, and $f = 0$ denotes normal data. In addition, the recognition rate is also used to analyze how well the method recognizes the working conditions. Some state-of-the-arts process monitoring methods are also introduced in this section for comparisons, including online dictionary learning (ODL) [26], PCA mixture model (mPCA) [27], k-nearest neighbor (KNN) [28], and SVM [29].

A. Numerical Simulation Experiment

A numerical simulation system has been designed to simulate the real industrial process. The data are generated by the following equation:

$$y = As + e \tag{23}$$

where y is the generated process data. $e = [e_1, e_2, e_3, e_4, e_5]^T$ denotes the observed noise in the process, which is usually assumed to be Gaussian white noise so that $e_i \sim N(0, 0.01^2)\forall i$. $s = [s_1, s_2]^T$ denotes the state vector of the process, setting different states to get data of different modes. A denotes the observation matrix, which has the values as follows:

$$A = \begin{bmatrix} 0.5768 & 0.3766 \\ 0.7382 & 0.0566 \\ 0.8291 & 0.4009 \\ 0.6519 & 0.2070 \\ 0.3972 & 0.8045 \end{bmatrix}. \tag{24}$$

The procedure of the proposed process monitoring method can be divided into three steps. First, training data and test data are generated by (23). Then, based on the training data, the simplified dictionary and classifier are trained according to Algorithm 1 and used as the edge layer model. Finally, the test data are monitored, including fault detection and working condition identification. The monitoring effect is analyzed by quantitative indicators mentioned previously.

The effect of model simplification on monitoring effectiveness is first tested experimentally. In this experiment, the state vectors are set as follows. For mode 1, set $s_1 \sim N(2, 1)$ and $s_2 \sim N(-3, 1)$. For mode 2, set $s_1 \sim N(-5, 1)$ and $s_2 \sim N(5, 1)$. We generate 1000 training data and 1000 test data for each mode, respectively. A bias fault of 0.2 is added to the first dimension of the last 500 test data for each mode. The parameters of the model are set as follows. The number of atoms of the cloud dictionary is 200. The coefficient α is 1 and β is 4. The sparsity for sparse

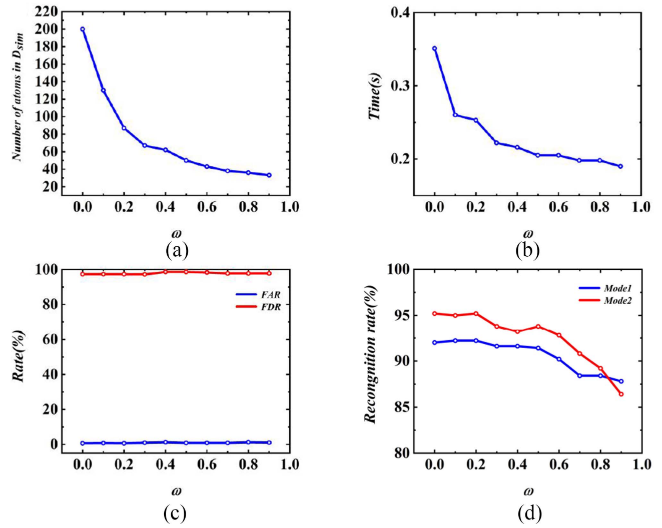

Fig. 3. Impact of dictionary simplification on monitoring effectiveness. (a) Effect of ω on the number of atoms in D_{sim}. (b) Effect of ω on the monitoring time. (c) Effect of ω on the FAR and FDR. (d) Effect of ω on the recognition rates.

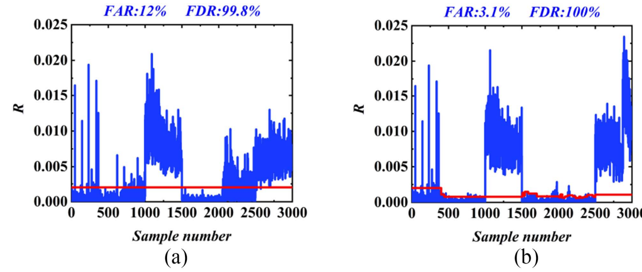

Fig. 4. Impact of the model update on monitoring effectiveness. (a) Monitoring effectiveness of model without update. (b) Monitoring effectiveness of the updated model.

TABLE I
QUANTITATIVE MONITORING RESULTS FOR MODEL UPDATE AND MODEL WITHOUT UPDATE

Model	FAR	FDR	Recognition rate of mode 1	Recognition rate of mode 2
Model without update	12%	99.8%	95.1%	75.2%
Model update	3.1%	100%	96.6%	95%

representation is 3. The confidence level for the control limit is 0.99.

During the model simplification stage, the coefficient ω is set from 0 to 0.9 for model simplification to obtain simplified models of different sizes. The indicators of monitoring the test data for each simplified model is recorded, including FAR, FDR, the recognition rate for both modes, and the time spent in monitoring the test data. The results of the experiment are shown in Fig. 3. The number of atoms in the simplified dictionary is sensitive to ω. A relatively small ω can thoroughly simplify the model. As the dictionary is simplified, the time taken to match atoms in the sparse representation is also reduced so that real-time monitoring is greatly improved. Meanwhile, although the model is simplified, the impact on the monitoring results is not significant. Low FAR, high FDR, and high recognition rate are guaranteed in most cases, and it is only when the model is severely simplified that the recognition rates drop more. This experiment shows the superiority of model simplification. A suitable ω can reduce the size of the dictionary and save time in monitoring, which can improve the real-time performance at the edge layer and achieve accurate monitoring.

Next, the numerical simulation experiment is used to simulate time-varying data by adding slow time variation to the state vectors. For mode 1, $s_1 = \cos(0.003i) + 2\sin(0.002i)$ and $s_2 \sim N(-3, 1)$. For mode 2, $s_1 = 2\cos(0.003i) + \sin(0.002i)$ and $s_2 \sim N(5, 1)$. i denotes the serial number of the sample. We produce 200 training data and 1500 test data for each mode, with a bias fault of 0.2 is added to the first dimension of the last 500 test data for each mode. The monitoring results of the model with and without update are compared. The parameters for the experiment are set the same as in the previous experiment. For the model update, the threshold σ is set to be 5, and μ is set to be 10. After training the model using the training data and simplifying it, the simplified model is used to monitor the test data and update the model when an update is triggered. The comparison between model update and without update is shown in Fig. 4 and Table I. The comparison results revealed that model mismatch can easily occur when the model is not updated. In particular, when monitoring data of mode 2, the model mismatch resulted in a reduced ability to represent normal data, with many normal data being misjudged as faults. Furthermore, according to the experimental results in Fig. 4, some of the normal data in mode 1 have reconstruction errors that exceeded the control limit. This is due to the model mismatch caused by the time-varying test data, which is eliminated as the model is updated again. In terms of indicators, this resulted in a large FAR and a greatly reduced recognition rate of mode 2. If the model was updated, this would not happen and all monitoring indicators improved. This experiment shows that model updating is necessary for time-varying industrial process monitoring.

Finally, the monitoring effectiveness of the proposed method to time-varying data is compared with other monitoring methods. The parameters for the comparison methods are set as follows. For ODL, the number of atoms is 80 and the sparsity is 3. For mPCA, the cumulative percent variance (CPV) is set to be 0.85. For a fair comparison, the confidence level for all methods is 0.99. For KNN, the value of k is 5. The SVM uses a linear kernel function. The monitoring results are shown in Fig. 5 and Table II. ODL is a process monitoring method that updates the model online. When monitoring the first mode, the ODL model represents the normal data well, but when the second mode appears, the constant updating of the model causes it to lose its ability to represent the normal data so that there are more false alarms for the normal data of mode 2. The mPCA is a method for monitoring multimode data, but it is not effective when the data are time varying. Although the proposed method has a slightly lower recognition rate than KNN and SVM for mode 1, the proposed method has the best results when both modes are considered. The proposed method does not only adapt to

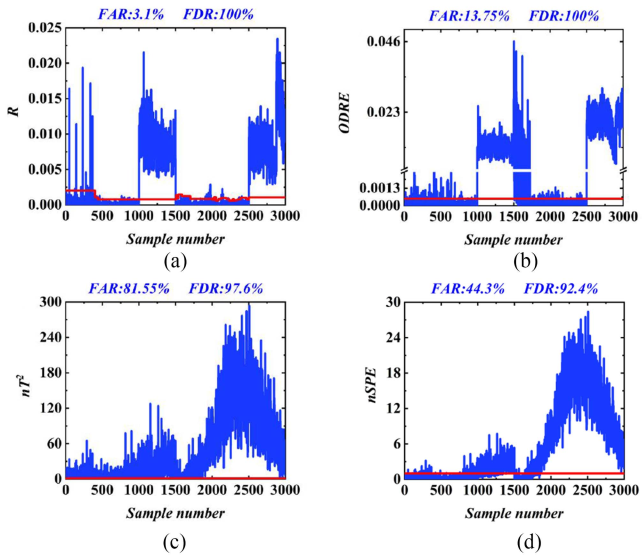

Fig. 5. Experiment results of the comparison methods on numerical simulation. (a) R statistic of the proposed method. (b) ODRE statistic of the ODL method. (c) nT^2 statistic of the mPCA method. (d) nSPE statistic of the mPCA method.

TABLE II
QUANTITATIVE MONITORING RESULTS FOR COMPARISON METHODS ON NUMERICAL SIMULATION

Method	FAR	FDR	Recognition rate of mode 1	Recognition rate of mode 2
The proposed method	3.1%	100%	96.6%	95%
ODL	13.75%	100%	—	—
mPCA(nT^2)	81.55%	97.6%	—	—
mPCA($nSPE$)	44.3%	92.4%	—	—
KNN	—	—	100%	86.8%
SVM	—	—	100%	83.7%

multimode data but also updates the model according to the time-varying data, thus providing the best monitoring results.

B. Industrial Roasting Process Results

To verify the feasibility of the proposed method for real industrial processes, we studied the roasting process at Zhuzhou Smelting Group in Hunan Province, China. Roasting is the first step in zinc smelting. The stable operation of the roaster is important to reduce pollution and improve the quality of the output zinc. Therefore, it is essential to monitor the faults and the real-time working conditions of the roaster to ensure its stable and healthy operation. The boiling roaster, IoT sensors, edge devices, and private cloud constitute the physical architecture of the cloud-edge collaborative monitoring framework, as shown in Fig. 1(a). An edge controller is deployed at the roaster site to handle the real-time data collected by the sensors and upload the data to the private cloud. The roaster's private cloud and the edge controller are located approximately 3 km apart. The cloud is responsible for managing large-scale data and algorithm models. The three layers and the data interaction between the layers constitute the network architecture of the cloud-edge collaborative framework, as shown in Fig. 1(b). The physical layer communicates with the edge layer via the OPC UA protocol. And the edge layer communicates with the cloud server via HTTP protocol. Meanwhile, the administrator can monitor and manage the system by interacting with the cloud server.

TABLE III
QUANTITATIVE MONITORING INDICATORS FOR COMPARISON METHODS ON ROASTING PROCESS

Method	FAR	FDR	Recognition rate of condition 1	Recognition rate of condition 2
The proposed method	0.38%	98.5%	99.75%	99.5%
ODL	1.63%	89.25%	—	—
mPCA(nT^2)	90.13%	100	—	—
mPCA($nSPE$)	1.5%	100%	—	—
KNN	—	—	94.5%	92%
SVM	—	—	100%	100%

Due to the limitations of the space, we build a roasting process cloud-edge collaborative framework in the laboratory to simulate the real roasting process site. The cloud platform is built in the Xinghuan cloud environment, which is provided by a Chinese Big Data company. The personal computer is used as the edge device. For the physical layer, we use a laboratory-created digital roaster, which is created by combining real roaster operation data and process mechanics. We generate 900 data of high-efficiency condition, 900 data of over-decomposition condition, and 400 data of under-oxidation condition by the digital roaster. The high-efficiency condition is used as normal condition 1, the over-decomposition condition as normal condition 2, and the under-oxidation condition as the faulty condition. 500 data in each normal condition are used as training data and the remaining 400 as the test data. In the test set, there are 200 faulty data after each normal condition. Therefore, there are a total of 1000 normal samples in the training set and a total of 1200 samples in the test set with 800 normal data and 400 faulty data.

An experiment is conducted to compare the monitoring indicators for all methods. For a fair comparison, the parameters of all methods are set as follows. For the proposed method, the number of atoms in the cloud dictionary is 100, the sparsity is 5, α is 4, β is 2, and ω is 0.2. The threshold σ for label error is 1.4 and μ is 10. For ODL, the number of dictionary atoms is 100 and the sparsity is 5. For mPCA, the CPV is 0.85. The confidence level for all fault detection methods is set to be 0.99. For the comparison methods of working condition recognition, the KNN has a k of 5 and the SVM uses a linear kernel function.

The monitoring results are shown in Fig. 6 and Table III. For the ODL method, there is a good FAR, but a poor FDR. This is because ODL may use faulty data incorrectly when updating, resulting in the reconstruction error of the faulty data less than the control limit. The monitoring effect of the nSPE statistic for mPCA is good, but the FAR of nT^2 statistic is poor. This is because PCA-based methods assume that the data obey the Gaussian distribution, but the data in the actual process may not satisfy this distribution, so the modeling effect is compromised. Among the methods for working condition recognition, the SVM achieves 100% recognition accuracy and the accuracy of the proposed method is slightly lower. This is because the proposed

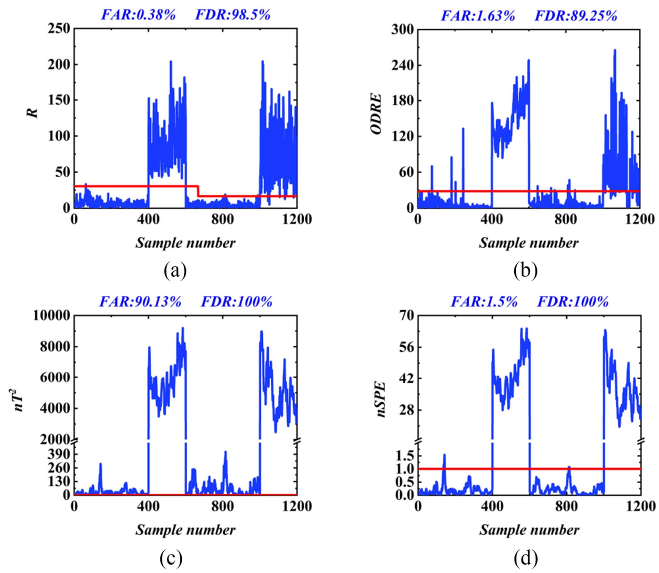

Fig. 6. Experiment results of the comparison methods on the roasting process. (a) R statistic of the proposed method. (b) Online dictionary reconstruction error (ODRE) statistic of the ODL method. (c) nT^2 statistic of the mPCA method. (d) n-squared prediction error (nSPE) statistic of the mPCA method.

method generates false alarms for a small number of normal condition data during fault detection, which has an impact on working condition recognition. However, the proposed method has the best process monitoring results for the roasting process when all indicators are considered.

IV. Conclusion

In this article, we focused on multimode industrial process monitoring based on the cloud-edge collaborative framework. The proposed method first used the dictionary learning model in the cloud to ensure the accuracy of monitoring. Then, the model was simplified and deployed to the edge for saving computing resources at the edge and guaranteeing real-time monitoring. Due to the time-varying feature of industrial data, an error-triggered strategy was proposed to update the process monitoring model in the cloud to solve the model mismatch puzzle. This article details the steps of cloud modeling, model simplification, error-triggered model update strategy, and edge layer online monitoring of the proposed method. Extensive numerical simulation and industrial roasting process results verified the superiority of model simplification and model update method. Compared with some state-of-the-arts process monitoring methods, the proposed method had better performance in fault detection and working condition recognition.

References

[1] B. Li, X. Chai, B. Hou, L. Zhang, J. Zhou, and Y. Liu, "New generation artificial intelligence-driven intelligent manufacturing (NGAIIM)," in *Proc. IEEE SmartWorld, Ubiquitous Intell. Comput., Adv. Trusted Comput., Scalable Comput. Commun., Cloud Big Data Comput., Internet People, Smart City Innov.*, 2018, pp. 1864–1869.

[2] J. Zhou, K. Athukorala, E. Gilman, J. Riekki, and M. Ylianttila, "Cloud architecture for dynamic service composition," *Int. J. Grid High Perform. Comput.*, vol. 4, no. 2, pp. 17–31, 2012.

[3] L. Zhang *et al.*, "Cloud manufacturing: A new manufacturing paradigm," *Enterprise Inf. Syst.*, vol. 8, no. 2, pp. 167–187, 2014.

[4] I. Ahmed, A. Dagnino, and Y. Ding, "Unsupervised anomaly detection based on minimum spanning tree approximated distance measures and its application to hydropower turbines," *IEEE Trans. Automat. Sci. Eng.*, vol. 16, no. 2, pp. 654–667, Apr. 2019.

[5] S. Yin, S. X. Ding, X. Xie, and H. Luo, "A review on basic data-driven approaches for industrial process monitoring," *IEEE Trans. Ind. Electron.*, vol. 61, no. 11, pp. 6418–6428, Nov. 2014.

[6] J. Zhu, Z. Ge, and Z. Song, "Distributed parallel PCA for modeling and monitoring of large-scale plant-wide processes with Big Data," *IEEE Trans. Ind. Informat.*, vol. 13, no. 4, pp. 1877–1885, Aug. 2017.

[7] G. Wang and S. Yin, "Quality-related fault detection approach based on orthogonal signal correction and modified PLS," *IEEE Trans. Ind. Informat.*, vol. 11, no. 2, pp. 398–405, Apr. 2015.

[8] X. Lu, W. Zou, and M. Huang, "A novel spatiotemporal LS-SVM method for complex distributed parameter systems with applications to curing thermal process," *IEEE Trans. Ind. Informat.*, vol. 12, no. 3, pp. 1156–1165, Jun. 2016.

[9] W. Yu and C. Zhao, "Recursive exponential slow feature analysis for fine-scale adaptive processes monitoring with comprehensive operation status identification," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3311–3323, Jun. 2019.

[10] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.

[11] Z. Jiang, Z. Lin, and L. S. Davis, "Label consistent K-SVD: Learning a discriminative dictionary for recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2651–2664, Nov. 2013.

[12] Q. Zhang and B. Li, "Discriminative K-SVD for dictionary learning in face recognition," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 2691–2698.

[13] Y. Lu and Y. Wang, "Machine fault diagnosis of fused filament fabrication process with physics-constrained dictionary learning," *Procedia Manuf.*, vol. 53, pp. 726–734, 2021.

[14] K. Huang, H. Wen, H. Ji, L. Cen, X. Chen, and C. Yang, "Nonlinear process monitoring using kernel dictionary learning with application to aluminum electrolysis process," *Control Eng. Pract.*, vol. 89, pp. 94–102, 2019.

[15] C. Yang *et al.*, "Multimode process monitoring based on robust dictionary learning with application to aluminium electrolysis process," *Neurocomputing*, vol. 332, pp. 305–319, 2019.

[16] C. Lai, W. Chien, L. T. Yang, and W. Qiang, "LSTM and edge computing for Big Data feature recognition of industrial electrical equipment," *IEEE Trans. Ind. Informat.*, vol. 15, no. 4, pp. 2469–2477, Apr. 2019.

[17] A. Caggiano, "Cloud-based manufacturing process monitoring for smart diagnosis services," *Int. J. Comput. Integr. Manuf.*, vol. 31, no. 7, pp. 612–623, 2018.

[18] W. Zhang *et al.*, "A streaming cloud platform for real-time video processing on embedded devices," *IEEE Trans. Cloud Comput.*, vol. 9, no. 3, pp. 868–880, Jul./Sep. 2011.

[19] L. Ren, Y. Liu, X. Wang, J. Lü, and M. J. Deen, "Cloud-edge based lightweight temporal convolutional networks for remaining useful life prediction in IIoT," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12578–12587, Aug. 2021.

[20] H. Huang, L. Yang, Y. Wang, X. Xu, and Y. Lu, "Digital twin-driven online anomaly detection for an automation system based on edge intelligence," *J. Manuf. Syst.*, vol. 59, pp. 138–150, 2021.

[21] Y. Wang, M. Liu, P. Zheng, H. Yang, and J. Zou, "A smart surface inspection system using faster R-CNN in cloud-edge computing environment," *Adv. Eng. Informat.*, vol. 43, 2020, Art. no. 101037.

[22] S. Lu, X. Tang, Y. Zhu, and J. She, "A cloud-edge collaborative intelligent fault diagnosis method based on LSTM-VAE hybrid model," in *Proc. 8th IEEE Int. Conf. Cyber Secur. Cloud Comput., 7th IEEE Int. Conf. Edge Comput. Scalable Cloud*, 2021, pp. 207–212.

[23] X. Zhang, Q. Liu, D. Wang, L. Zhao, and S. Maybank, "Self-taught semi-supervised dictionary learning with non-negative constraint," *IEEE Trans. Ind. Informat.*, vol. 16, no. 1, pp. 532–543, Jan. 2020.

[24] S. K. Sahoo and A. Makur, "Signal recovery from random measurements via extended orthogonal matching pursuit," *IEEE Trans. Signal Process.*, vol. 63, no. 10, pp. 2572–2581, May 2015.

[25] Z. Ge and Z. Song, "Process monitoring based on independent component analysis- principal component analysis (ICA- PCA) and similarity factors," *Ind. Eng. Chem. Res.*, vol. 46, no. 7, pp. 2054–2063, 2007.
[26] K. Huang *et al.*, "Adaptive process monitoring via online dictionary learning and its industrial application," *ISA Trans.*, vol. 114, pp. 399–412, 2021.
[27] X. Xu, L. Xie, and S. Wang, "Multi-mode process monitoring method based on PCA mixture model," *CIESC J.*, vol. 62, no. 3, pp. 743–752, 2011.
[28] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, "KNN model-based approach in classification," in *Proc. OTM Confederated Int. Conf. Move Meaningful Internet Syst.*, 2003, pp. 986–996.
[29] N. Cristianini *et al.*, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge, U.K.: Cambridge Univ. Press, 2000.

Keke Huang received the B.A. degree in automatic control from Northeastern University, Shenyang, China, in 2012, and the Ph.D. degree in control science and engineering from Tsinghua University, Beijing, China, in 2017.

From April 2017 to September 2021, he was an Associate Professor with Central South University, Changsha, China, where he is currently a Full Professor with the School of Automation. His research interests include industrial Big Data, process monitoring and control, and network sciences.

Zui Tao received the B.A. degree in automation, in 2021, from Central South University, Changsha, China, where he is currently working toward the M.A. degree in control science and engineering.

His research interests include dictionary learning and process monitoring.

Chen Wang received B.S. and M.S. degrees in computer science from Fudan University, Shanghai, China in 2003 and 2006, respectively. He is currently working toward the D.Eng. (part-time) in software engineering from Tsinghua University, Beijing, China.

He is currently a Senior Researcher and CTO with National Engineering Laboratory for Big Data Software, Tsinghua University. Before this, he was a Senior Manager and Research Staff Member with Information Management Research Group, IBM Research—China. He is serving as the Head of Industrial Big Data Working Group, National IT Standardization Committee (TC28), China and the Deputy Secretary-General of Alliance of Industrial Internet, China. His research interests include industrial Big Data and data management.

Tianxu Guo received the B.E. degree in automation from Northeastern University, Shenyang, China, in 2012, and the Ph.D. degree in control science and engineering from Tsinghua University, Beijing, China, in 2019.

He is currently conducting postdoctoral research with the Chinese Academy of Fiscal Sciences, Beijing. His research interests include data-driven process monitoring with application in industry and economy.

Chunhua Yang (Fellow, IEEE) received the M.S. degree in automatic control engineering and the Ph.D. degree in control science and engineering from Central South University, Changsha, China, in 1988 and 2002, respectively.

Since 1999, she has been a Full Professor with the School of Information Science and Engineering, Central South University, where she is currently the Head of Department with the School of Automation. Her current research interests include modeling and optimal control of complex industrial processes, and intelligent control systems.

Weihua Gui received the B.Eng. degree in electrical engineering and the M.S. degree in automatic control engineering from Central South University, Changsha, China, in 1976 and 1981, respectively.

Since 2013, he has been an Academician with the Chinese Academy of Engineering. He is currently with the School of Automation, Central South University. His current research interests include modeling and optimal control of complex industrial processes, fault diagnoses, and distributed robust control.