Process Safety and Environmental Protection 168 (2022) 953–970

Contents lists available at ScienceDirect

Process Safety and Environmental Protection

journal homepage: www.journals.elsevier.com/process-safety-and-environmental-protection

Dynamic-scale graph neural network for fault detection

Zhengqing Lin, Zhengwei Hu, Jingchao Peng, Haitao Zhao *

Automation Department, School of Information Science and Engineering, East China University of Science and Technology, Shanghai 200237, PR China

ARTICLE INFO

Keywords:
Process monitoring
Fault detection
Dynamic feature extraction
Dimension reduction

ABSTRACT

Traditional graph-based dynamic fault detection methods describe the dynamic characteristic through constructing a single neighborhood graph at the current sample with some history samples. However, they ignore the diversity of dynamic properties of the variables in complex chemical processes. To overcome this problem, a novel neural network structure combining multiscale subgraphs is proposed, named dynamic-scale graph neural network (DSGNN), which divides variables into multiple groups according to their dynamic properties. DSGNN constructs a subgraph in each group. In traditional graph-based methods, the scale of the graph is usually manually designed. In DSGNN, the scale of each subgraph is decided by the dynamic properties of the variables in this subgraph. To aggregate the dynamic information, DSGNN utilizes convolution operations. The weights assigned to the neighbors in each subgraph are determined according to the similarity between the current data and its neighbors. Low-dimensional features are extracted through the back-propagation technique from the updated high-dimensional features produced by convolution operations. Two case studies on a multivariate dynamic process and the Tennessee Eastman process are conducted to show the superiority of DSGNN.

1. Introduction

In modern industry, safety is a continuing concern. Despite advances in the industry that have made production more intelligent, real-world processes are often complicated and inevitably in a fault condition, which results in shut-downs, injuries, and economic losses (Venkatasubramanian et al., 2003). It is imperative to achieve high levels of safety with intelligent process monitoring technology.

More and more attention has been given to process monitoring from the perspective of safety and risk management. Khan et al. (2015) reviewed the development of process safety in terms of risk management (RA) and illustrated that the integration of fault detection and diagnosis (FDD) and RA improved the level of process safety. Arunthavanathan et al. (2021) further expanded the connotation of FDD, RA, and abnormal situation management (ASM) from safety perspectives. Yao et al. (2022) proposed a dynamic artificial immune system based on simulated vaccine and correlation coefficient methods (SV-CCDAIS) to improve the safety of chemical systems under the extreme absence of data. Yin et al. (2021) proposed a data mining method for the risk assessment and smart alert of gas kick during the industrial drilling process. Kopbayev et al. (2022) proposed a model for gas leakage detection. The convolutional network (to model spatial likelihood of leak) is combined with the bidirectional long short-term memory layer network (to model temporal dependence of leak likelihood) to perform leak detection and diagnosis. In summary, an effective monitoring model can reduce the risk of accidents and enhance process safety.

Process monitoring methods can be divided into three categories: (1) model-based, (2) knowledge-based (3) data-based methods (Jiang et al., 2019; Han et al., 2022). On the one hand, the structure of industrial chemical processes becomes more and more complex. Obtaining an accurate process model and monitoring results using model-based and knowledge-based methods is difficult. On the other hand, with the quick utilization of advanced measure technologies in chemical processes, a large amount of data can be collected and stored in a database (Luo et al., 2014). Data-based methods have developed rapidly.

As the fundamental analytical algorithms, principal component analysis (PCA) (Amin et al., 2021; Harrou et al., 2016), partial least squares (PLS) (Fazai et al., 2019; Tahir et al., 2019), and independent component analysis (ICA) (Li et al., 2017; Li and Yan, 2019) keep motivating novel approaches for fault monitoring. Amin et al. (2021) proposed a framework by integrating PCA with the Bayesian network. Harrou et al (Harrou et al., 2016). combined the multivariate exponentially weighted moving average (MEWMA) monitoring scheme with PCA to enhance anomaly detection performance. The success of these methods mentioned above is based on the assumption that the observations at one time are statistically independent of observations at past

* Corresponding author.
E-mail address: haitaozhao@ecust.edu.cn (H. Zhao).

https://doi.org/10.1016/j.psep.2022.10.036
Received 13 June 2022; Received in revised form 26 September 2022; Accepted 17 October 2022
Available online 19 October 2022

times. For typical chemical processes, this assumption is valid only for long sampling intervals. Shorter sampling intervals are desired for speedy fault detection, in which case a method taking into account the serial correlations in the data may improve fault detection (Russell et al., 2000). The dynamic behavior varied according to a specific process, may include useful process information, and can be utilized to improve accuracy in fault detection (Rong et al., 2012).

To capture the dynamic information between samples for process fault detection, many methods have been reported. Dynamic principal component analysis (DPCA) (Ku et al., 1995) incorporates dynamic properties by applying PCA to an extended data matrix that contains current as well as history samples. Choi et al (Choi and Lee, 2004). proposed dynamic kernel principal component analysis (DKPCA) based on DPCA to capture the dynamic information in nonlinear process data. Besides, some deep learning methods are widely used in dynamic fault detection (Jiang et al., 2017; Zhao, 2018; Hu et al., 2021). Jiang et al (Jiang et al., 2017). proposed the dynamic stacked sparse autoencoder (DSSAE) which combines the dynamic data window with the stacked sparse autoencoder (SSAE). Zhao et al (Zhao, 2018). adopted augmented matrices instead of extended vectors to capture the dynamic information and the neighborhood information. In these methods mentioned above, the current sample is concatenated with several neighbors to construct extended vectors or matrices to capture the dynamic information. Assume the number of neighbors is l, and the dimensionality of the extended vector is l times that of the original data. To determine the number of neighbors, the autocorrelation function (ACF) is commonly used (Zhao, 2018; Hu et al., 2021; Liu et al., 2022; Soltanalian and Stoica, 2012). Autocorrelation is a mathematical tool that can measure the cross-correlation of a variable with itself at different time points. If significant autocorrelation is shown in the autocorrelation chart, the dynamic problem can be fully considered in fault detection. However, the extended vector or matrices enlarge the dimension of the original data, which is prone to cause the so-called curse of dimensionality problem (Hu et al., 2021).

Recently, a number of graph-based methods have been proposed and applied for dynamic fault detection. These methods do not enlarge the dimension of the original data space, avoiding the curse of dimensionality problem. Traditional graph-based methods include locally linear embedding (LLE) (Zhang et al., 2017), locality preserving projection (LPP) (Deng and Tian, 2013), neighborhood preserving embedding (NPE) (Song et al., 2014), etc. To handle the dynamic process monitoring problem, Miao et al. (2013) put forward the time neighborhood preserving embedding model (TNPE) to describe the dynamic characteristic. The dynamic variations are maintained by reconstructing each data sample from its time sequence adjacent neighbors. Yang et al. (2017) proposed a new method called time information constrained embedding (TICE), using a new expression of time weight to qualify the importance of sequential neighbors. Liu et al. (2022) utilized graph convolution operations to aggregate information from neighbors to update the information of the current. Li et al. (2020) incorporated the horizontal visibility graph (HVG) and graph neural network (GNN) for dynamic bearing faults diagnosis. Tong et al. (2021) presented a transient fault detection and classification approach in power transmission lines based on the graph convolutional neural network. However, these methods mentioned above only construct one graph with a number of neighbors assuming that different variables perform similarly in the dynamic aspect. Dynamic autocorrelation of the whole measured variables is considered, but dynamic autocorrelation of each variable is not analyzed (Yin and Yan, 2019). For example, variables such as the pressure of the reactor or the temperature of the reactor have long time delays in processes, which means these variables have stronger serial autocorrelations between samples. Some variables such as the level of the reactor have no time delays in processes, which means these variables have no serial autocorrelations. Constructing only one graph with a number of neighbors leads to the loss of structural information between variables. Besides, the scale of the neighborhood graph is selected manually in traditional methods, which can be further improved.

In this paper, a novel graph-based method named DSGNN is proposed and applied to dynamic process monitoring. DSGNN utilizes ACF to calculate the autocorrelation of variables and measure their dynamic properties. According to the results of ACF, the variables are divided into groups by the hierarchical clustering method. The operation aims to make the variables having similar dynamic properties considered simultaneously. DSGNN constructs a subgraph in each group and the variables in the same group are considered as a subsample. The dimensionality of the subsample is the number of these variables. Then, to aggregate the dynamic information, DSGNN utilizes convolution operations in each subgraph. It is inappropriate to treat neighbors equally, thus the weight assigned to each neighbor is not the same. In each subgraph, the weights are determined based on the similarity between the current subsample and its neighbors. It is worth noting that DSGNN does not change the dimensionality of the data, avoiding the curse of dimensionality problem. Besides, compared with traditional methods which design the scale of the neighborhood graph manually, the scale of each subgraph in DSGNN is determined by the dynamic properties of the subsample. The features generated by convolution operations are used as the input of neural network structure to extract the low-dimensional features. Finally, T^2 and SPE statistics are constructed to realize fault detection. Compared with traditional linear autoregressive models (ARMA, ARMAX, etc), DSGNN is a nonlinear model with two neural network structures, which means that DSGNN has a stronger learning ability in nonlinear chemical processes. Besides, ARMA considers the dynamic information of the whole measured variables and ignores the dynamic autocorrelation of each variable. DSGNN constructs dynamic-scale subgraphs taking the diversity of the dynamic properties of variables into account and capturing the structure information between variables. The main contribution and innovation can be summarized as follows:

- A novel fault detection method called DSGNN is put forward. DSGNN captures the dynamic characteristics of the data in complex chemical processes. The neural network structure is used to extract nonlinear features based on dynamic-scale subgraphs.
- The dynamic property of each variable is analyzed. Some variables have strong autocorrelation, but some have no or weak autocorrelation. DSGNN decides the scale of the subgraphs according to the dynamic property of the variables.
- Compared with traditional graph-based dynamic fault detection methods, DSGNN constructs dynamic-scale subgraphs taking the local structure information between variables into account and obtaining better monitoring performance.

The reminder of this paper is as follows: Section 2 introduces the structure of the time neighborhood preserving embedding method (TNPE) and dynamic stacked autoencoder (DSAE). Section 3 describes the structure of the proposed DSGNN. Section 4 illustrates the fault detection based on the DSGNN. Section 5 utilizes a numerical example and the Tennessee Eastman process (TEP) to demonstrate the superiority of the DSGNN. Section 6 summarizes the work of this paper.

2. Related work

2.1. Time neighborhood preserving embedding

The neighborhood graph in time neighborhood preserving embedding (TNPE) is constructed by the time sequence adjacent points. The dynamic relationship can be captured by exploring the geometric properties of each neighborhood. By minimizing the squared reconstruction error between all the data points with their reconstructions as Eq. (1), the reconstruction weight matrix W is obtained. Finally, the low-dimensionality features are extracted by the linear transformation operations. Received the l dimensional input data containing n samples

$X = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n) \in \mathbb{R}^{l\times n}$. The whole procedures are as follows:

(1) Construct the neighborhood graph of each data point $\boldsymbol{x}_i$ associated with its $2K$ adjacent neighbors: $N_i \in \{\boldsymbol{x}_{i-k}, \ldots, \boldsymbol{x}_{i-1}, \boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_{i+k}\}$. As shown in Fig. 1, the current sample is located in the middle graph, the fixed time span contains the corresponding $2K$ nearest neighbors. It should be noted that the future samples are included in the graph. When a test sample arrives, this sample will not be fed into the model until K future samples are provided. This operation is a trade-off between efficiency and accuracy, which is commonly used in graph-based fault detection methods such as (Liu et al., 2022; Miao et al., 2013; Yang et al., 2017).

(2) Reconstruct each data point from its neighbors. Presuming each time neighborhood distributes linearly, we compute the reconstruction weight matrix W by minimizing

$$Loss = \sum_i \left\| \boldsymbol{x}_i - \sum_j W_{ij}\boldsymbol{x}_j \right\|^2 \tag{1}$$

with two constraints: each sample $\boldsymbol{x}_i$ is reconstructed only from its neighbors, which means $W_{ij} = 0$ if $\boldsymbol{x}_j$ does not belong to the time neighbors of $\boldsymbol{x}_i$; the sum of the rows of the weight matrix is equal to 1. W_{ij} reflects the intrinsic dynamic structure information of the data.

(3) Compute the transformation matrix A. The objective of the method is to get a transformation matrix $A = (\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_d) \in \mathbb{R}^{l\times d}$ which transforms high-dimensional data X to low-dimensional feature $Y = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n) \in \mathbb{R}^{d\times n}$, namely $\boldsymbol{y}_i = A^T\boldsymbol{x}_i$. Based on the geometric intuition that the dynamic variations of the original data manifold can be optimally preserved in the low-dimensional manifold, the transformation matrix A can be calculated by minimizing

$$\sum \left\| \boldsymbol{y}_i - \sum_j W_{ij}\boldsymbol{y}_i \right\|^2 = \sum \left\| A^T\boldsymbol{x}_i - \sum_j W_{ij}A^T\boldsymbol{x}_i \right\|^2 \tag{2}$$

Calculate the smallest d eigenvalues of the eigenvalue problem $XMX^T\boldsymbol{a} = \lambda XX^T\boldsymbol{a}$, and the transformation matrix A is constructed by the eigenvectors corresponding to the eigenvalues, where $M = (I - W)^T(I - W)$.

2.2. Dynamic stacked autoencoder

In this section, we give a brief introduction to DSAE. The training data can be denoted as $\boldsymbol{x}_i = [x_{i1}, x_{i2}, \ldots, x_{il}]^T \in \mathbb{R}^{l\times 1}(i = 1, 2, \ldots n)$, where x_{ij} means the jth sensor measurement value at time i. To deal with the dynamic information between the samples, the extended vectors as Eq. (3) are the input of classical stacked autoencoder:

$$\boldsymbol{x}_{extend_i} = [\boldsymbol{x}_{i-m+1}, \boldsymbol{x}_{i-m+2}, \ldots, \boldsymbol{x}_i]^T \in \mathbb{R}^{(lm)\times 1}(i = m, m+1, \ldots n) \tag{3}$$

where m means the lagged number of the extended vectors. Stacking multiple autoencoders constitutes a stacked autoencoder (SAE). The extended vectors are the input of SAE. For the SAE stacked by two basic autoencoders, the features extracted by the first AE are used as input for the second AE. The first AE aims at reconstructing the raw input. Then the second AE aims at reconstructing the output of the hidden layer of the first AE.

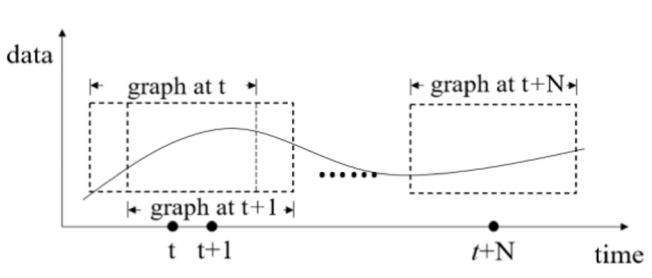

Fig. 1. Graphs at different time points.

The parameters of the first AE can be represented as $[W^{(1,1)} \in \mathbb{R}^{p\times(lm)}, \boldsymbol{b}^{(1,1)} \in \mathbb{R}^p]$ and $[W^{(1,2)} \in \mathbb{R}^{(lm)\times p}, \boldsymbol{b}^{(1,2)} \in \mathbb{R}^{(lm)}]$. The input $\boldsymbol{x}_{extend_i} \in \mathbb{R}^{(lm)\times 1}$ is reconstructed and the output data can be represented as $\widehat{\boldsymbol{x}}_{extend_i} \in \mathbb{R}^{(lm)\times 1}$. The encoding process is represented as:

$$z = g_1(\boldsymbol{x}) = \sigma(W^{(1,1)}\boldsymbol{x}_{extend_i} + \boldsymbol{b}^{(1,1)}) \tag{4}$$

The decoding process from the hidden layer to the output layer is represented as:

$$\widehat{\boldsymbol{x}}_{extend_i} = g_2(\boldsymbol{z}) = \sigma(W^{(1,2)}\boldsymbol{z} + \boldsymbol{b}^{(1,2)}) \tag{5}$$

where $z \in \mathbb{R}^{p\times 1}$ denotes the output of the hidden layer of the first AE. Then z is reconstructed by the second AE. The encoding process of the second AE is represented as:

$$\boldsymbol{h} = g_3(\boldsymbol{z}) = \sigma(W^{(2,1)}\boldsymbol{z} + \boldsymbol{b}^{(2,1)}) \tag{6}$$

The decoding process is represented as:

$$\widehat{z} = g_4(\boldsymbol{h}) = \sigma(W^{(2,2)}\boldsymbol{h} + \boldsymbol{b}^{(2,2)}) \tag{7}$$

where $[W^{(2,1)} \in \mathbb{R}^{h\times p}, \boldsymbol{b}^{(2,1)} \in \mathbb{R}^h]$ and $[W^{(2,2)} \in \mathbb{R}^{p\times h}, \boldsymbol{b}^{(2,2)} \in \mathbb{R}^p]$, $\sigma(\bullet)$ denote the sigmoid function and $\boldsymbol{h} \in \mathbb{R}^{h\times 1}$ denotes the output of the hidden layer.

The training procedure contains two steps: pre-training and fine-tuning. The pre-training is layer-by-layer greedy training which means that only one AE with one hidden layer is trained at a time. The fine-turning is required to train the network parameters to be globally optimal. The parameters obtained in the pre-training phase are used as initialization parameters for this phase.

3. Dynamic-scale graph neural network

Motivated by TNPE and DSAE, this paper designs a nonlinear method to incorporate both the nonlinear information and dynamic information between samples. Section 3.1 introduces autocorrelation function (ACF) which is used to measure the dynamic properties of variables. Section 3.2 introduces the whole structure of DSGNN as shown in Fig. 3.

3.1. Autocorrelation function

The successful discovery of the dynamic relationship in TNPE is based on the assumption that the dynamic properties of variables are approximate. This is often not the case. To better illustrate the diversity of the dynamic properties of the observations, we utilize the autocorrelation function (Johnson et al., 2014).

Autocorrelation is a mathematical tool that can measure the cross-correlation of a variable with itself at different time points. Let ξ denote a stochastic process, ξ_t denotes the sample at t moment. The calculate equation for the autocorrelation of ξ_t and ξ_{t+k} where $k = 0, 1, \ldots, K$ is:

$$\widehat{r}_k = \frac{c_k}{c_0} \tag{8}$$

where $c_k = \frac{1}{T-1}\sum_{t=1}^{T-k}(\xi_t - \overline{\xi})(\xi_t - \overline{\xi})$, $\overline{\xi}$ denotes the mean of $\xi_t (t = 0, 1, \ldots, T)$, $c_0$denotes the sample variance of the process. The calculate equation of the standard error for testing the significance of a single lag-m autocorrelation is:

$$SE_r = \sqrt{\frac{(1 + 2\sum_{i=1}^{m-1}\widehat{r}_i^2)}{N}} \tag{9}$$

Fig. 2 shows the autocorrelation of four variables in Tennessee Eastman process (TEP) (Downs and Vogel, 1993) which are measured by the autocorrelation function (ACF). As shown with the blue lines, the 95% confidence intervals are drawn at $\pm 2SE_r$. Fig. 2(a) shows that the reactor pressure has positive autocorrelation in 15 time lags. In Fig. 2(b), it can be found that the dynamic property of reactor temperature is weaker than reactor pressure. It has negative autocorrelation in 5 time lags. Fig. 2(c) illustrates that compressor recycle flow has no serial correlation, thus it is reasonable to consider compressor recycle flow as an independent variable. Fig. 2(d) shows that the reactor cool temperature has positive autocorrelation in only 5 time lags. Constructing only one graph cannot characterize the dynamic properties of different variables well. For example, a model with a graph containing 10 neighbors is unable to fully capture the dynamic information of variables that perform strong autocorrelation in 15 time lags. But the model creates redundant information for variables that perform autocorrelation in only 5 time lags. It is reasonable to group these variables according to their dynamic performance. A subgraph with a property amount of past and future data points can be constructed in each group. Besides, the separation of positive and negative autocorrelation groups can better characterize the dynamic relationship between samplesFig. 3.

3.2. The structure of DSGNN

In DSGNN, variables are divided into multiple groups: no autocorrelation, strong positive autocorrelation in relatively short time lags, strong positive autocorrelation in relatively middle time lags, strong positive autocorrelation in relatively long time lags, strong negative autocorrelation in relatively short time lags, in relatively middle time lags and in relatively long time lags. The number of short time lags, middle time lags and long time lags in subgraphs of variables having positive autocorrelation can be denoted as a_p, b_p, c_p, where $(a_p < b_p < c_p)$. The number of short time lags, middle time lags and long time lags in subgraphs of variables having negative autocorrelation can be denoted as a_n, b_n, c_n, where $(a_n < b_n < c_n)$. Given a data sample $\boldsymbol{x}_i \in \mathbb{R}^{l\times 1}$, seven groups are obtained according to ACF: $\{\boldsymbol{x}_{i_1} \in \mathbb{R}^{l_1\times 1}, \boldsymbol{x}_{i_2} \in \mathbb{R}^{l_2\times 1}, \ldots, \boldsymbol{x}_{i_7} \in \mathbb{R}^{l_7\times 1}\}$ where $l = l_1 + l_2 + l_3 + l_4 + l_5 + l_6 + l_7$. The neighbor numbers of the subgraphs can be denoted as $\{N_{i_1}, N_{i_2}, \ldots, N_{i_7}\}$. The details of these subgraphs are illustrated in Table 1. With these subgraphs, the local structure information between variables can be captured. It should be noted that the values of the short time lags, middle time lags, long time lags, and the number of groups are not fixed because the dynamic properties of variables in different industrial processes are not consistent. The procedures of getting the grouping results and the values of the neighbor numbers can be summarized as:

- Calculate the autocorrelation of all variables and obtain the autocorrelation value n of each variable which means this variable has strong autocorrelation in n time lags.
- Utilizing the hierarchical cluster method to group variables.
- Construct a subgraph in each group. The variables in a group form a subsample. The dimensionality of the subsample is the number of variables. Calculate the autocorrelation of the subsample as (Liu et al., 2022), namely, calculate the autocorrelation of the sum of the variables. Then the number of neighbors is the same as the autocorrelation of the subsample.

To extract the dynamic information from the time sequence neighbors in each subgraph, instead of concatenating the current data with neighbors, we utilize the convolution operations and assign a weight to each neighbor to update the current data. As shown in Fig. 1, the current sample is located in the middle subgraph, and the fixed time span contains the corresponding neighbors. In this way, the dimensionality of the data does not change, avoiding the curse of dimensionality problem. It

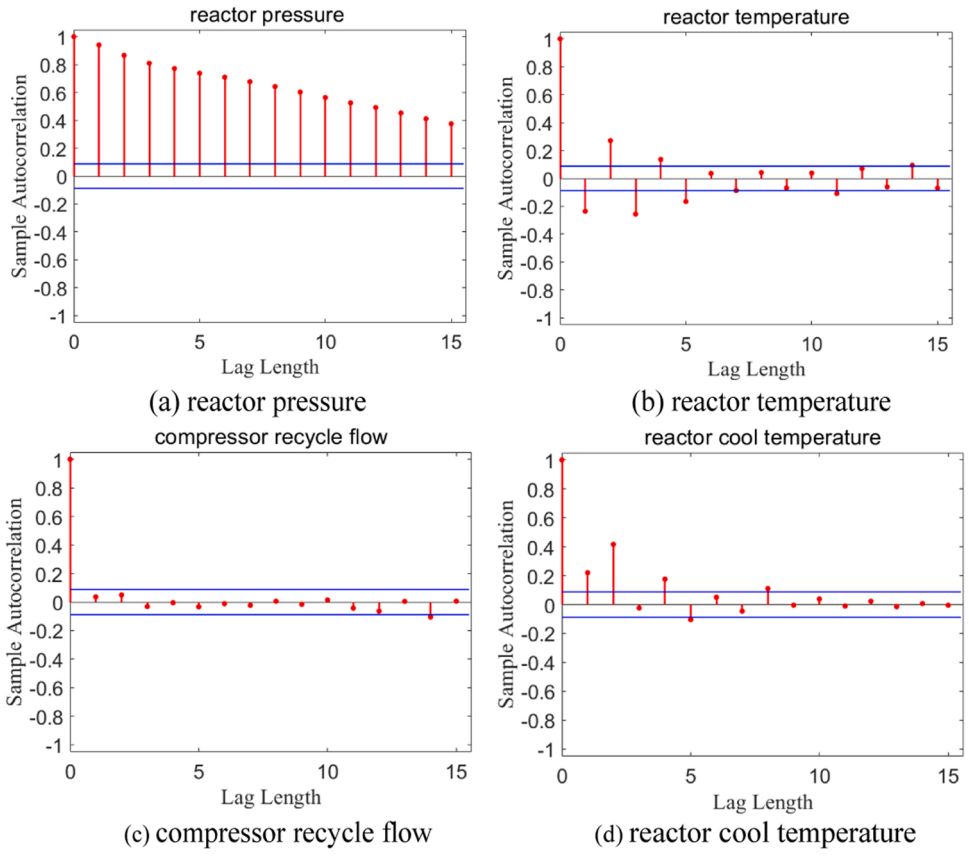

Fig. 2. Number of time lags determined from autocorrelation function of different measurements.

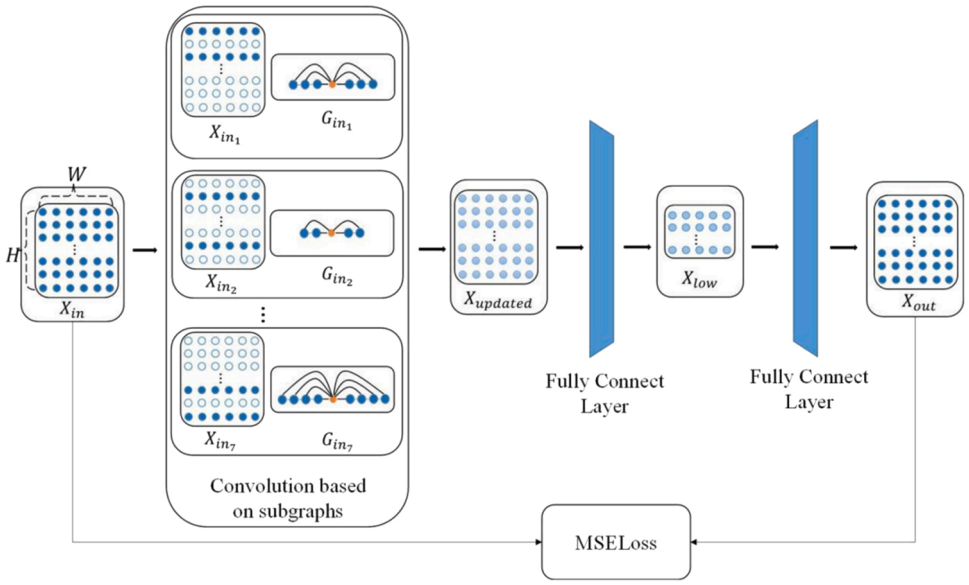

Fig. 3. The structure of DSGNN. W denotes the number of samples, H means the dimension of a sample. X_{in} is the input matrix, X_{in_i} is the ith section of the input matrix. G_{in_i} is the subgraph constructed from the ith section of the input matrix. In X_{in_i}, the hollow circles denote the data which are not included in G_{in_i} and the dark blue solid circles denote the data contained in G_{in_i}.

Table 1
The details of subgraphs.

Autocorrelation	Time lags	Nodes number	Weight
No	0	0	–
Positive	a_p	a_p	Euclidean distances normalized by softmax
Negative	a_n	a_n	Cosine similarity
Positive	b_p	b_p	Euclidean distances normalized by softmax
Negative	b_n	b_n	Cosine similarity
Positive	c_p	c_p	Euclidean distances normalized by softmax
Negative	c_n	c_n	Cosine similarity

should be noted that when test data arrives, this sample will not be fed into the model until the future samples are provided. This operation is a trade-off between efficiency and accuracy.

Different neighbors have different degrees of influence on the current data. The weights assigned to the neighbors with higher autocorrelation can be larger, while the weights assigned to the neighbors with lower autocorrelation can be smaller.

For those subgraphs where variables perform positive autocorrelation, we calculate the weights by the Euclidean distance normalized by softmax between the current data and neighbors. Let $\boldsymbol{x}_{i_m}$ and $\boldsymbol{x}_{j_m}$ represent the mth section of the current data and a neighbor where $m \in \{1, 2, \ldots, 7\}$, we define the weight between these two samples as:

$$dist_{i_m j_m} = \left\{ \begin{array}{c} \left\| \boldsymbol{x}_{i_m} - \boldsymbol{x}_{j_m} \right\|_2^2 \text{if} \boldsymbol{x}_{j_m} \quad \text{is a neighbor of} \boldsymbol{x}_{i_m} \\ 0 \text{otherwise} \end{array} \right\} \tag{10}$$

$$W_{i_m j_m} = \left\{ \begin{array}{c} 1 \text{if} i_m = \quad j_m \\ softmax(-\frac{dist_{i_m j_m}}{\beta_{i_m}}) \text{if} \boldsymbol{x}_{j_m} \quad \text{is a neighbor of} \boldsymbol{x}_{i_m} \\ 0 \text{otherwise} \end{array} \right\} \tag{11}$$

where β_{i_m} means the sum of the Euclidean distances of the neighbors. The values of the weights range from 0 to 1. The more similar the neighbors are with the current data, the closer the weights are to 1. On the contrary, the less similar the neighbors are with the current data, the closer the weights are to 0.

As for the subgraphs where variables perform negative autocorrelation, we choose the Cosine similarity between the mth section of the current data $\boldsymbol{x}_i$ and a neighbor $\boldsymbol{x}_j$ as the weight:

$$W_{i_m j_m} = \left\{ \begin{array}{c} \frac{\boldsymbol{x}_{i_m} \bullet \boldsymbol{x}_{j_m}}{|\boldsymbol{x}_{i_m}| \times |\boldsymbol{x}_{j_m}|} \text{if} \boldsymbol{x}_{j_m} \quad \text{is a neighbor of} \boldsymbol{x}_{i_m} \\ 0 \text{otherwise} \end{array} \right\} \tag{12}$$

The values range from − 1–1. The more similar the neighbors are with the current data, the closer the weights are to 1. On the contrary, the less similar the neighbors are with the current data, the closer the weights are to − 1.

For the subgraph of the section $\boldsymbol{x}_{i_m}$ where $m \in \{1, 2, \ldots, 7\}$, its $2K$ adjacent neighbors:$N_{i_m} \in \{\boldsymbol{x}_{i_m - k_m}, \ldots, \boldsymbol{x}_{i_m - 1}, \boldsymbol{x}_{i_m + 1}, \ldots, \boldsymbol{x}_{i_m + k_m}\}$. The updated features $\boldsymbol{x}_{updated} \in \mathbb{R}^{l \times 1}$ can be concatenated by 7 sections: $\{\boldsymbol{x}_{updated_1} \in \mathbb{R}^{l_1 \times 1}, \boldsymbol{x}_{updated_2} \in \mathbb{R}^{l_2 \times 1}, \ldots, \boldsymbol{x}_{updated_7} \in \mathbb{R}^{l_7 \times 1}\}$, the section $\boldsymbol{x}_{updated_m}$ can be calculated as follows:

$$\boldsymbol{x}_{updated_m} = \sum_{k=-k_m}^{k_m} W_{i_m (i+k)_m} \boldsymbol{x}_{(i+k)_m} \tag{13}$$

the dynamic message is maintained in the updated features. The updated features are the input of two neural network structures. The neural network structures can be denoted as layer α and layer β respectively:

$$[\alpha, \beta] = \underset{\alpha, \beta}{\operatorname{argmin}} \left\| \boldsymbol{x}_i - \beta(\alpha(\boldsymbol{x}_{updated})) \right\|^2 \tag{14}$$

where $\boldsymbol{x}_i \in \mathbb{R}^{l \times 1}$ denotes the input data. Layer α transforms the updated features $\boldsymbol{x}_{updated}$ to the low-dimensional features $\boldsymbol{x}_{low} \in \mathbb{R}^{h \times 1}$:

$$\boldsymbol{x}_{low} = \alpha(\boldsymbol{x}_{updated}) = \sigma(W^{\alpha} \boldsymbol{x}_{updated} + \boldsymbol{b}^{\alpha}) \tag{15}$$

where $\sigma(\bullet)$ means the sigmoid function. W^{α} is a parameter matrix. $\boldsymbol{b}^{\alpha}$ is an offset vector. Layer β transforms the low-dimensional features $\boldsymbol{x}_{low} \in \mathbb{R}^{h \times 1}$ to the reconstructed output $\boldsymbol{x}_{out}$ as the same dimension as $\boldsymbol{x}_i$:

$$\boldsymbol{x}_{out} = \beta(\boldsymbol{x}_{low}) = \sigma(W^{\beta} \boldsymbol{x}_{low} + \boldsymbol{b}^{\beta}) \tag{16}$$

To learn the optimal parameters $[W^{\alpha}, \boldsymbol{b}^{\alpha}; W^{\beta}, \boldsymbol{b}^{\beta}]$, the back-propagation technology is utilized.

4. Fault detection based on DSGNN

In this section, a new dynamic fault detection method based on DSGNN is proposed. Firstly, we collect normal process data $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{l \times n}$. Secondly, we divide normal data into groups according to ACF: $\{X_{i_1} \in \mathbb{R}^{l_1 \times n}, X_{i_2} \in \mathbb{R}^{l_2 \times n}, \ldots, X_{i_7} \in \mathbb{R}^{l_7 \times n}\}$ where $l = l_1 + l_2 + l_3 + l_4 + l_5 + l_6 + l_7$. Through convolution operations, the updated features $X_{updated} = [\boldsymbol{x}_{u_1}, \boldsymbol{x}_{u_2}, \ldots, \boldsymbol{x}_{u_n}] \in \mathbb{R}^{l \times n}$ containing dynamic information are produced. Thirdly, the low-dimensional global features as $X_{low} = [\boldsymbol{x}_{low_1}, \boldsymbol{x}_{low_2}, \ldots, \boldsymbol{x}_{low_n}] \in \mathbb{R}^{h \times n}$ are obtained. Finally, Hotelling $T^2$and the squared prediction error (*SPE*) statistics are calculated for fault detection based on the low-dimensional features.

The T^2 and *SPE* statistics of $\boldsymbol{x}_{low_i} \in \mathbb{R}^h$ are defined as:

$$T_i^2 = \boldsymbol{x}_{low_i}^T S^{-1} \boldsymbol{x}_{low_i} \tag{17}$$

$$SPE_i = \left\| \boldsymbol{x}_i - \sigma(W^\beta \sigma(W^\alpha \boldsymbol{x}_{u_i} + \boldsymbol{b}^\alpha) + \boldsymbol{b}^\beta) \right\|^2 \tag{18}$$

where S is the covariance matrix. Kernel density estimation (KDE) (Samuel and Cao, 2016) is used to determine the confidence limits of T^2 and *SPE*. Firstly, the T^2 statistics $[T_1^2, T_2^2, \ldots, T_n^2]$ of the normal process data $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{l \times n}$ is calculated by:

$$\widehat{f}(T^2) = \frac{1}{n\gamma} \sum_{i=1}^{n} K\left(\frac{T^2 - T_i^2}{\gamma}\right) \tag{19}$$

where $K(\bullet)$ means a kernel function. We take the Gaussian kernel for density estimation, which is given by

$$\widehat{f}(T^2) = \frac{1}{\sqrt{2\pi}n\gamma} \sum_{i=1}^{n} exp\left(\frac{(T^2 - T_i^2)^2}{2\gamma^2}\right) \tag{20}$$

where γ represents a bandwidth parameter. The confidence limit T_c at significant level α is estimated by:

$$\int_0^{T_c} \widehat{f}(T^2) dT^2 = 1 - \alpha \tag{21}$$

The confidence limit SPE_c of *SPE* statistic is calculated in the same way as the T^2 statistic. And if the T^2 statistic or *SPE* statistic of a new sample exceeds the limit, it means that the process fault is detected.

4.1. Monitoring procedure

The offline modeling and online monitoring flowchart are shown in Fig. 4. The offline modeling and online monitoring procedures are summarized as follows:

Algorithm 1. The offline modeling and online monitoring procedures.

Stage I: Offline modeling.

Step1: Normalize the training samples X to zero mean and unit variance.

Step2: Divide the variables into groups utilizing the hierarchical

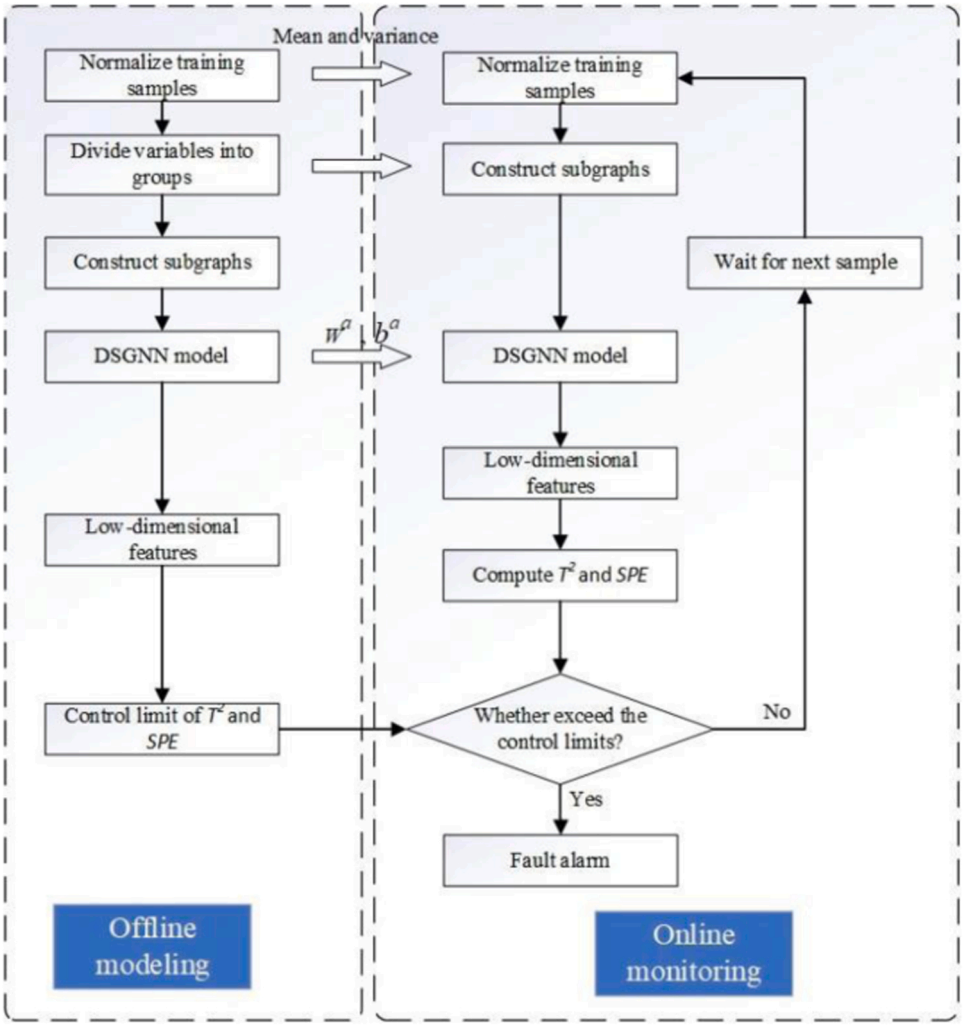

Fig. 4. Flowchart of DSGNN for fault detection.

cluster method.

Step3: Construct dynamic-scale subgraphs.

Step4: Calculate the updated features based on these subgraphs.

Step5: Train the DSGNN model and obtain the optimal parameters and the low-dimensional features.

Step6: Calculate the T^2 and the *SPE* statistics and their control limits.

Stage II: Online monitoring.

Step1: Collect the test samples $\boldsymbol{x}_{test}$. Normalize the test samples $\boldsymbol{x}_{test}$ using the parameters of the training samples X.

Step2: Obtain the updated test features $\boldsymbol{x}_{utest}$ according to the subgraphs constructed by the training samples X.

Step3: Obtain the low-dimensional features using the trained DSGNN model.

Step4: Determine the T^2 and the *SPE* statistics of the low-dimensional features.

Step5: Alarm if T^2 exceeds the control limit T^2_{limit}or *SPE* exceeds the control limit SPE_{limit}; otherwise, consider $\boldsymbol{x}_{test}$ as a normal process data.

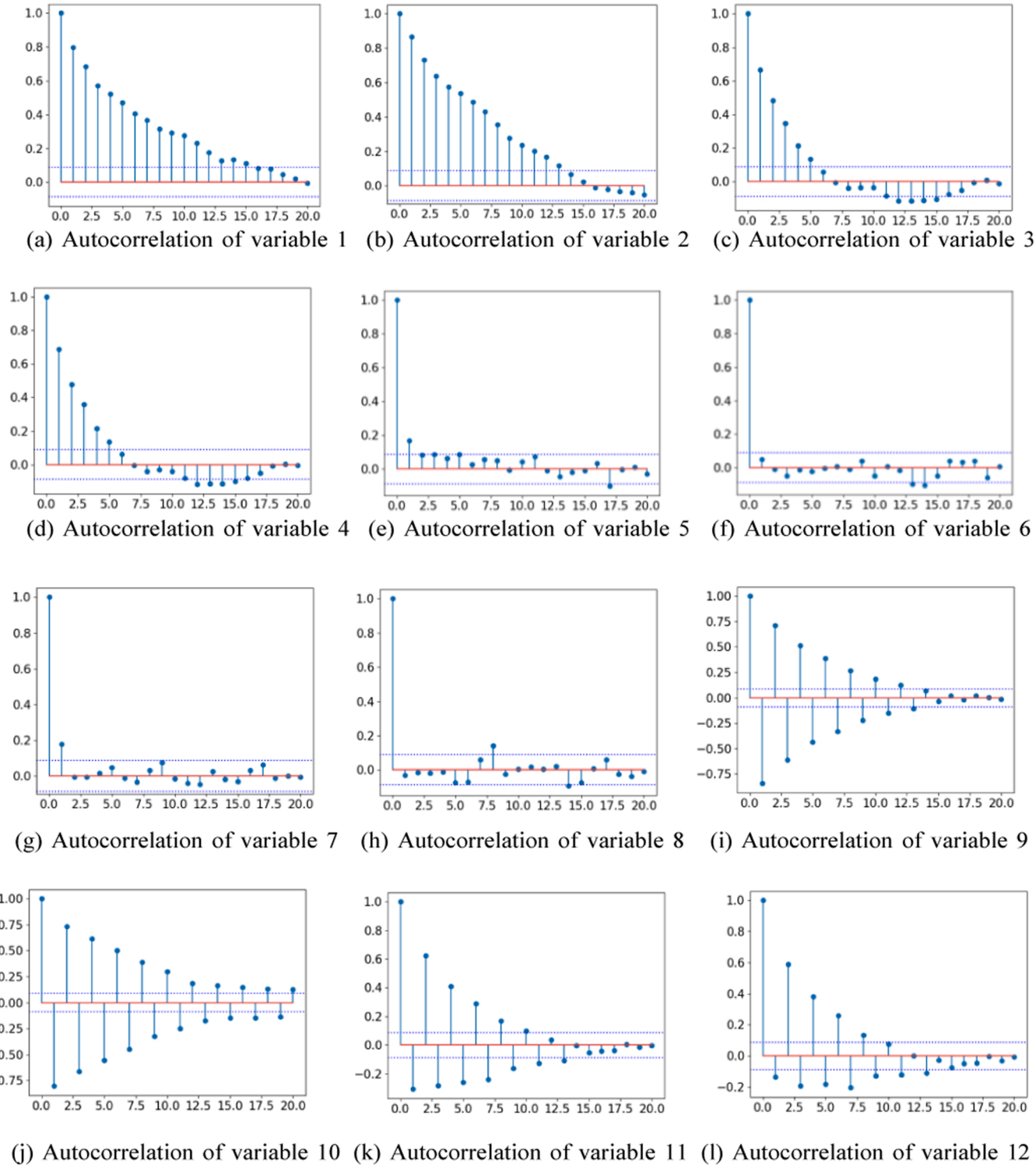

Fig. 5. The autocorrelation of the variables.

5. Experiments and discussion

To demonstrate the effectiveness of DSGNN method, we employ two cases, a nonlinear dynamic numerical case and the Tennessee Eastman process.

5.1. Numerical example

The samples $\widehat{x}_i = [\widehat{x}_{i1}, \widehat{x}_{i2}, \ldots, \widehat{x}_{i12}]^T$ modified by the literature (Zhu et al., 2020) are generated as follows:

$$\left\{\begin{array}{c} \widehat{x}_{i1} = u_i + e_1 \\ \widehat{x}_{i2} = u_{i-1}^2 - 2 \times u_{i-3} + e_2 \\ \widehat{x}_{i3} = u_{i-1}^2 + 0.5 \times v_{i-2} + e_3 \\ \widehat{x}_{i4} = v_{i-1}^2 + v_{i-2} + e_4 \\ \widehat{x}_{i5} = sin(u_{i-2}) + e_5 \\ \widehat{x}_{i6} = sin(u_{i-1}^2 - 2 \times u_{i-3}) + e_6 \\ \widehat{x}_{i7} = sin(u_{i-2} + u_{i-3}) + e_7 \\ \widehat{x}_{i8} = sin(u_{i-2} + 2 \times u_{i-3}) + e_8 \\ \widehat{x}_{i9} = 0.9 \times v_i + e_9 \\ \widehat{x}_{i10} = v_{i-1} + 0.3 \times v_{i-2} + e_{10} \\ \widehat{x}_{i11} = 0.05 \times v_i^2 + 0.3 \times v_{i-1} + e_{11} \\ \widehat{x}_{i12} = 0.05 \times v_i^2 + 0.25 \times v_{i-1} + e_{12} \end{array}\right. \tag{22}$$

where i means the sampling time, the noises are represented by $\{e_1, e_2, \ldots, e_{12}\}$, which are independently sampled from the normal distribution $\mathcal{N}(0, \quad 0.1)$. To ensure the dynamic properties of the data, we define u_i, v_i, $\varphi(\bullet)$, and $\psi(\bullet)$ as:

$$u_i = \varphi(u_{i-1}) + p_i$$

$$v_i = \psi(v_{i-1}) + q_i$$

$$\varphi(u_{i-1}) = \left\{\begin{array}{cc} u_{i-1} - 1 & if \quad u_{i-1} \geq 1 \\ 0.5 \times u_{i-1} + 1 & if \quad u_{i-1} \leq 1 \\ -0.8 \times u_{i-1} + 0.5 & otherwise \end{array}\right\}$$

$$\psi(v_{i-1}) = -0.9 \times v_{i-1} + 0.8 \tag{23}$$

where p_i and q_i are sampled from the distribution $\mathcal{N}(0, \quad 0.1)$. $\varphi(\bullet)$ and $\psi(\bullet)$ functions are used to achieve dynamic relationships between samples. To emphasize the diversity of the dynamic properties, the variables generated by Eq. (22) have different autocorrelation as shown in Fig. 5. Variable 1 and Variable 2 have strong positive autocorrelation in 15 time lags as shown in Fig. 5(a) and Fig. 5(b). Variable 3 and Variable 4 have strong positive autocorrelation in 5 time lags as shown in Fig. 5(c) and (d). Fig. 5(e), (f), (g), and (h) illustrate that Variable 5, Variable 6, Variable 7, and Variable 8 have weak serial correlations. Variable 9 and Variable 10 have strong negative autocorrelation in 15 time lags, and Variable 11 and Variable 12 have strong negative autocorrelation in 10 time lags. Training data and test data include 500 samples. The test data is simulated with a step fault: a step change of 1 is added to variable 5 from the 201st sample, and a step change of 2 is added to variable 4 from the 201st sample.

Fig. 7 shows the visualization results of the extracted features in two-dimensional spaces by TNPE, DKPCA, DSAE, DNOM (Hu et al., 2021), GDAE (Liu et al., 2022), and DSGNN. DKPCA denotes Dynamic kernel principal component analysis. DNOM is a novel nonlinear dynamic method, which consists of data dynamic extension, nonlinear feedforward neural network, and an orthogonal mapping matrix. Through backpropagation and Eigen decomposition (ED), DNOM can extract key low-dimensional features from the original high-dimensional data. GDAE is a nonlinear dynamic fault detection method based on graph neural network structure. But GDAE only constructs one graph and ignores the diversity of dynamic properties of variables. In GDAE, the number of neighbors is determined by the dynamic property of the sum of all the variables. We calculate the autocorrelation of the sum of the variables as in Fig. 6. We set the lagged number l= 10 in TNPE, DKPCA, DSAE, DNOM, and GDAE. In DSGNN, variables can be divided into five groups according to their dynamic properties. Ignoring Variable 5, Variable 6, Variable 7, and Variable 8 with weak serial correlations, four subgraphs are constructed: a 15 neighbors subgraph with Euclidean distances normalized by softmax as the weight for Variable 1 and Variable 2; a 5 neighbors subgraph with Euclidean distances normalized by softmax as the weight for Variable 3 and Variable 4; a 15 neighbors subgraph with Cosine similarity as the weight for Variable 9 and Variable 10; a 10 neighbors subgraph with Cosine similarity as the weight for Variable 11 and Variable 12.

In Fig. 7(b), (c), and (d), normal data and fault data are largely mixed up indicating that DKPCA, DSAE, and DNOM are not suitable for this process. TNPE, GDAE, and DSGNN perform better than DKPCA, DSAE, and DNOM illustrating the advantage of graph-based dynamic fault detection methods. In Fig. 7(a) and (e), normal and fault samples are partially separated and partially mixed. The degree of separation of GDAE is deeper than that of TNPE, illustrating that GDAE has better learning ability than TNPE in complex multivariable processes. DSGNN can clearly distinguish normal samples from fault samples and obtains the best results compared with the other methods.

5.2. Experiments on Tennessee Eastman process

Tennessee Eastman process (TEP) is a simulation of an actual chemical process that is proposed by Downs and Vogel (1993). It is widely used as a source of data for evaluating different fault detection methods. TEP has five operating units including a reactor, a condenser, a compressor, a stripper, and a vapor liquid separator. The overall structure is shown in Fig. 8.

In the experiment, a sample contains 52 variables including 22 continuous process variables, 19 combinations, and 11 manipulated variables. The descriptions of these variables can be found in Tables 2, 3, and 4. 21 fault data sets are generated which are shown in Table 5. The training data set contains 960 samples and the faults start at the 161st sample.

5.2.1. Parameters setting and analysis

We compare the proposed DSGNN method with TNPE, DKPCA, DSAE, DNOM, and GDAE. To determine the lagged number of the extended vectors or the number of neighbors in TNPE, DKPCA, DSAE, DNOM, and GDAE, we calculate the autocorrelation of the sum of all the variables in TEP as shown in Fig. 6. Thus we set the lagged number of the extended vectors or the number of neighbors of the graph l= 10. For dynamic fault detection methods such as DKPCA, DSAE, and DNOM, the

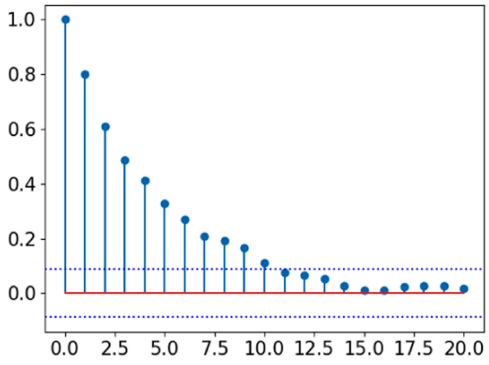

Fig. 6. The autocorrelation of the sum of the variables.

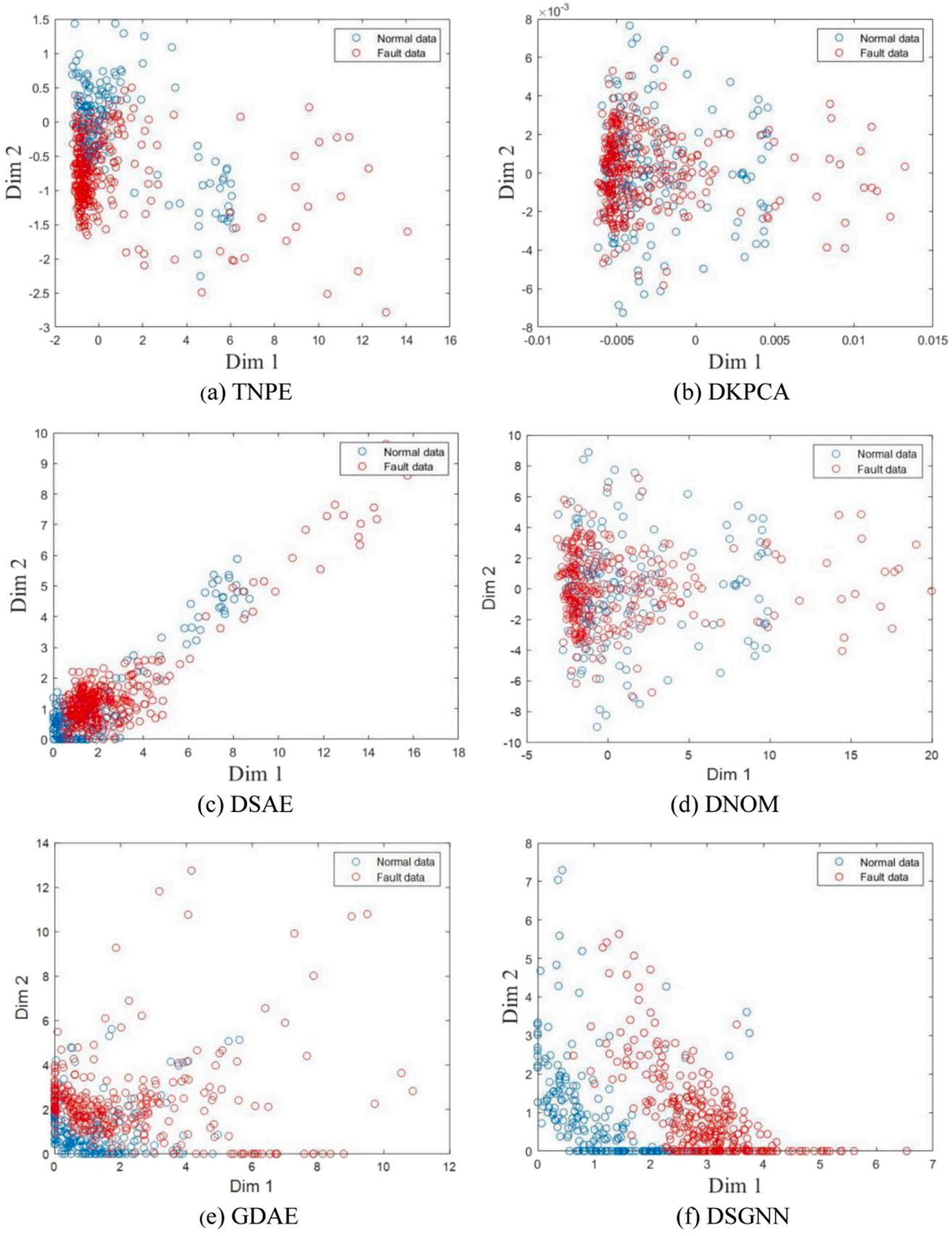

Fig. 7. the visualization results of extracted features on the first two dimensions by TNPE, DKPCA, DSAE, DNOM, GDAE, and DSGNN. Red dots and blue dots indicate fault data and normal data, respectively.

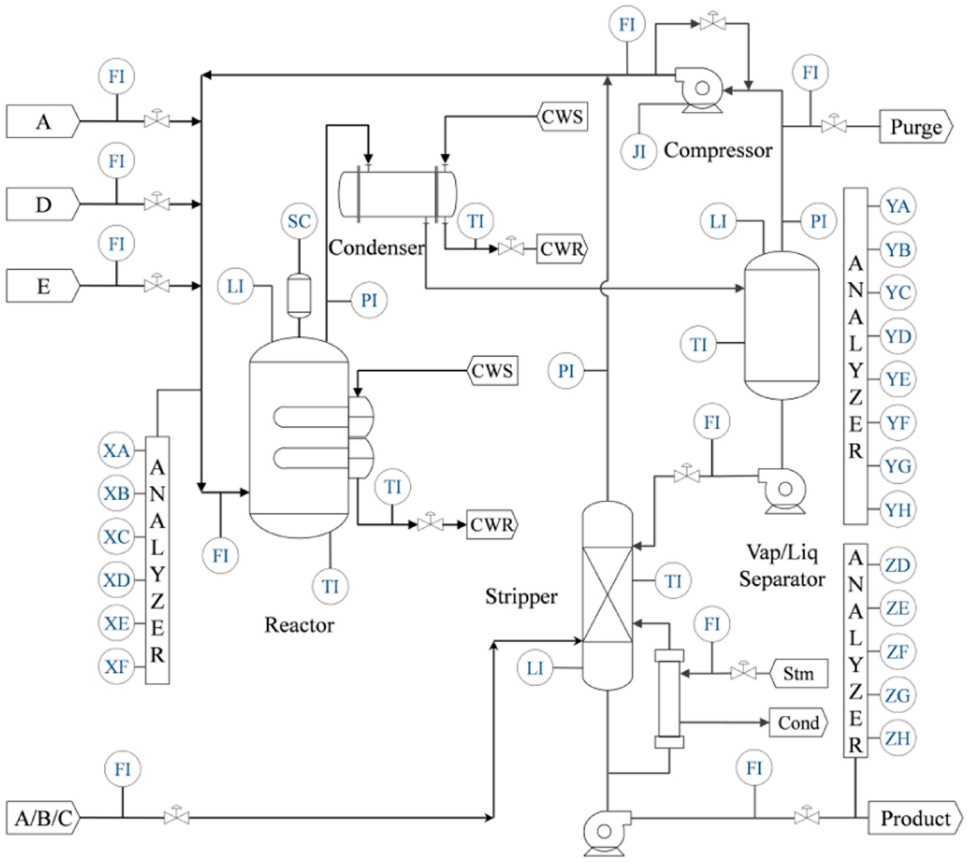

Fig. 8. A diagram of the TEP simulator.

Table 2
The descriptions of process variables in TEP.

No.	Process Variable	No.	Process Variable
1	A Feed	12	Separator Level
2	D Feed	13	Separator Pressure
3	E Feed	14	Separator Underflow
4	A+C Feed	15	Stripper Level
5	Recycle Flow	16	Stripper Pressure
6	Reactor Feed	17	Stripper Underflow
7	Reactor Pressure	18	Stripper Temp
8	Reactor Level	19	Steam Flow
9	Reactor Temp	20	Compressor Work
10	Purge Rate	21	Reactor Cool Temp
11	Separator Temp	22	Condenser Cool Temp

Table 3
The descriptions of component variables in TEP.

No.	Component Variable
23–28	A, B, C, D, E, F Feed
29–36	A, B, C, D, E, F, G, H Purge
37–41	D, E, F, G, H Product

input dimensionalities are 520. Aiming at preserving 85% information of the eigenspectrum (the sum of eigenvalues), we design the reduced dimensionality δ in DKPCA, DSAE, and DNOM to be 128 based on the cumulative percentage variance (CPV) rule. For the graph-based methods TNPE, DSGNN, and GDAE, since the input dimensionality does not change, the reduced dimensionalities are set to 27.

The kernel of DKPCA is a Gaussian kernel:

Table 4
The descriptions of manipulated variables in TEP.

No.	Manipulated Variable	No.	Manipulated Variable
42	D Feed	48	Separator Valve
43	E Feed	49	Stripper Valve
44	A Feed	50	Steam valve
45	A+C Feed	51	Reactor Coolant
46	Recycle Valve	52	Condenser Coolant
47	Purge Valve		

$$k(x_i, x_j) = exp(-\frac{\|x_i - x_j\|_2^2}{\eta}) \tag{24}$$

the kernel parameter η is $15ml\bar{\delta}$, where $\bar{\delta}$ means the average of the standard deviations of different variables (Zhao et al., 2006) and ml denotes the data dimension after using the data augmentation technique.

In DSGNN, Layer α transforms the updated features into low-dimensional features, and the node number of Layer α is the same as the reduced dimensionality $\delta = 27$. Layer β transforms the low-dimensional features to the reconstructed output, the node number of Layer β is 52. The learning rate is set to 0.005, and the number of iterations is set to 60. Adam method (Kingma and Ba, 2015) is used to optimize the parameters of DSGNN. The experiments are performed on a computer with specifications as follows: CPU: Intel (R) Core (TM) i7–9750 H CPU@ 2.60 GHZ, GPU: NVIDIA Geforce GTX 1050.

To obtain the grouping results and determine the neighbor numbers in each graph in DSGNN, the autocorrelation values must be obtained. The results are shown in Table 6. The value n means the variable has strong autocorrelation in n time lags. The positive values indicate

Table 5
TEP fault modes.

Fault No	Description	Type
1	A/C feed ratio, B composition constant (Stream 4)	Step
2	B composition, A/C ratio constant (Stream 4)	Step
3	D feed temperature (Stream 2)	Step
4	Reactor cooling water inlet temperature	Step
5	Condenser cooling water inlet temperature	Step
6	A feed loss (Stream 1)	Step
7	C header pressure loss (Stream 4)	Step
8	A, B, C feed composition (Stream 4)	Random variation
9	D feed temperature (Stream 2)	Random variation
10	C feed temperature (Stream 4)	Random variation
11	Reactor cooling water inlet temperature	Random variation
12	Condenser cooling water inlet temperature	Random variation
13	Reaction kinetics	Slow drift
14	Reactor cooling water valve	Sticking
15	Condenser cooling water valve	Sticking
16	Unknown	Unknown
17	Unknown	Unknown
18	Unknown	Unknown
19	Unknown	Unknown
20	Unknown	Unknown
21	Valve (Stream 4)	Constant position

positive autocorrelation, the negative values indicate negative autocorrelation.

The hierarchical cluster method is utilized to obtain the grouping results based on the results in Table 6. Variables with similar dynamic properties can be grouped together. Five groups are generated as shown in Figs. 9 and 10. In each group, we construct a subgraph. The variables having similar dynamic properties are considered as a subsample in the group. The number of neighbors is determined by the dynamic property of the subsample. The autocorrelation of the subsample is calculated by ACF as (Liu et al., 2022), namely, calculating the autocorrelation of the sum of the variables in this graph. The results of the autocorrelation of the subsamples in graphs are shown in Table 7. Then we set $a_p = 1, b_p = 5, c_p = 10, d_p = 15, b_n = 5, a_n = c_n = 0$ (Ignoring the variables which have no autocorrelation). For the subgraph consisting of only one variable where the variable performs positive autocorrelation, the weight of the neighbor is calculated by the Euclidean distance between the current data and the neighbor.

5.2.2. Case studies

The performance is compared with TNPE, DKPCA, DSAE, DNOM, and GDAE. The measurements are the missed detection rate (MDR), and false alarm rate (FAR). MDR means the ratio of the fault samples which are misidentified as normal samples when monitoring. FAR denotes the ratio of the normal samples which are misidentified as fault samples. We only consider that the fault is successfully detected when MDR $<$ 50% and FAR $<$ 5%. If MDR is larger than 50%, the detection performance is even worse than a random guess. When FAR $<$ 5%, MDR with smaller values indicates better performance. The test results of the 21 faults of the six methods are shown in Table 8. It should be noted that the neural network structure tends to approach the local minimum in the training process, so the results of DSAE, GDAE, DNOM, and GDAE in each experiment are slightly different. The results are obtained by averaging

Table 6
The autocorrelation of variables in TEP.

No.	Autocorrelation	No.	Autocorrelation	No.	Autocorrelation	No.	Autocorrelation
1	6	14	0	27	1	40	4
2	1	15	0	28	4	41	4
3	1	16	15	29	15	42	1
4	7	17	0	30	6	43	1
5	0	18	15	31	15	44	6
6	0	19	15	32	1	45	0
7	15	20	15	33	15	46	15
8	0	21	4	34	6	47	15
9	-5	22	4	35	2	48	0
10	7	23	4	36	2	49	0
11	15	24	5	37	10	50	15
12	0	25	15	38	15	51	-6
13	15	26	1	39	4	52	0

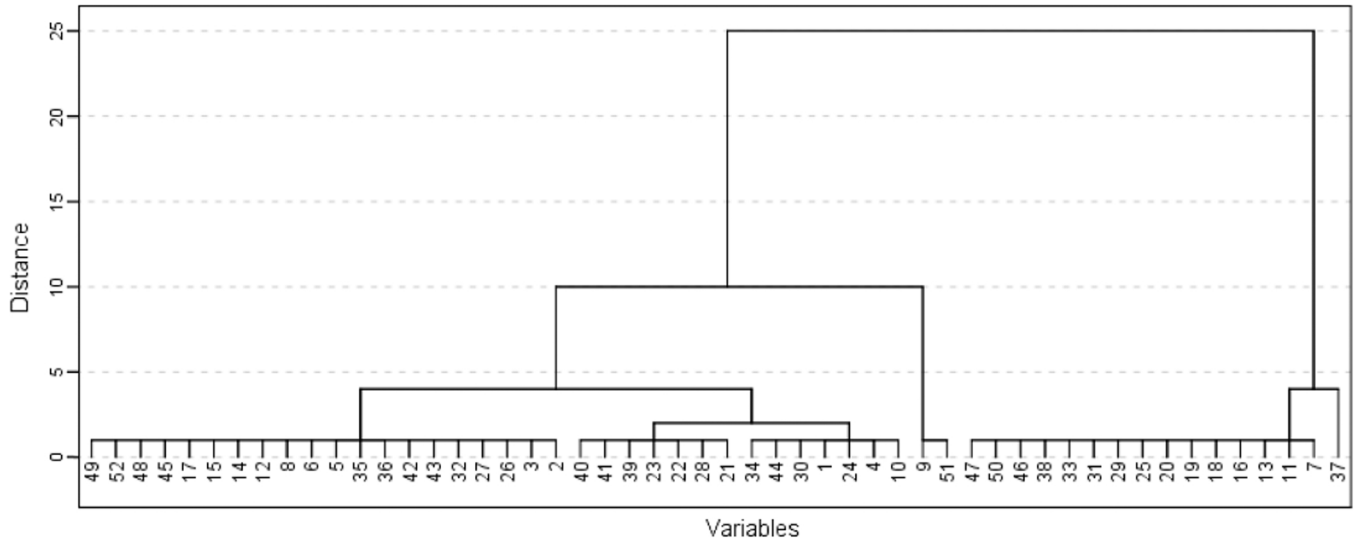

Fig. 9. Dendrogram of hierarchical clusters.

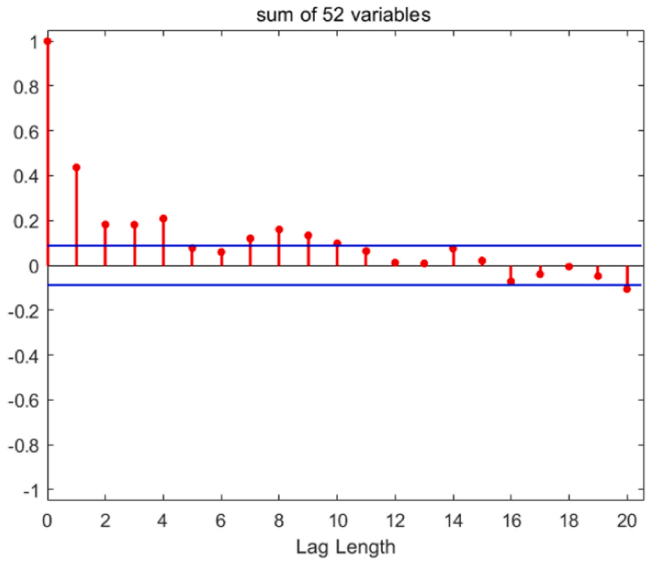

Fig. 10. The autocorrelation of the sum of the variables in TEP.

Table 7
The grouped results of the variables in TEP.

No.	Variables	Autocorrelation of the subsample
1	2,3,26,27,32,42,43,35,36	1
2	1,4,10,21,22,23,24,28,34,39,40,41,44	5
3	37	10
4	9,51	5
5	7,11,13,16,18,19,20,25,29,31,33,38,46,47,50	15

over 10 times of experiments to avoid randomness. The best detection result for each fault is highlighted in bold.

As shown in Table 8, the faults cannot be fully detected by any one of the six methods. DSGNN obtains the lowest MDRs in 11 cases, TNPE gives 7 best cases, DKPCA gives 5 best cases, DSAE gives 5 best cases, DNOM gives 6 best cases, and GDAE gives 8 best cases. TNPE, GDAE, and DSGNN perform better than DKPCA, DSAE, and DNOM illustrating the advantage of graph-based dynamic fault detection methods. TNPE is a linear method and its learning ability is limited to complex chemical processes. GDAE is a dynamic fault detection method based on graph neural network structures. But it only constructs a graph with a number of neighbors, ignoring the diversity of the dynamic properties of variables. DSGNN combines the strength of graph-based strategy with the neural network structure, emphasizes the diverse dynamic properties of variables, and obtains better results.

Figs. 11, 12, and 13 show the monitor results for Fault 11, Fault 13, and Fault 17. In these plots, the blue lines denote the 160 normal samples; the red lines represent the fault samples; the black dotted lines mean the control limit; the black arrows marked DT indicate the detection times. When blue lines are above the control limit, it means a false alarm. When red lines are below the control limit, it means missed detection.

In Fig. 11, we can find that TNPE, DKPCA, DSAE, and DNOM cannot detect Fault 11 correctly while the abnormal data and the normal data can be distinguished by the GDAE and DSGNN. From Table 8, the MDRs of T^2 statistic of TNPE, DKPCA, DSAE, and DNOM are 89.25%, 84.63%, 74.21%, and 87.25%, respectively. Values greater than 50% mean that the detection results are worse than random choice. The MDRs of T^2 statistic of GDAE and DSGNN are both 25%, which are much better than the other four methods. The detection times of GDAE and DSGNN are close to the detection times of traditional dynamic methods such as DSAE, DKPCA, and DNOM even when K future samples are added. It can be seen in Table 5 that Fault 11 is related to the reactor cooling water inlet temperature, meaning that there is a strong autocorrelation between samples, and this autocorrelation is not a simple linear relationship. It is critical to make full use of dynamic information to detect Fault 11. The results indicate that graph convolution is more suitable for processing dynamic data than constructing extended vectors and simply concatenating samples.

In Fig. 12, the MDRs of T^2 statistic of TNPE, DKPCA, DSAE, DNOM, GDAE, and DSGNN are 4.49%, 4.5%, 4.85%, 4.92%, 4%, and 3.88%. TNPE, GDAE, and DSGNN perform better than DKPCA, DSAE, and DNOM, indicating that the graph-based methods have better learning ability than traditional dynamic fault detection methods. TNPE is a linear method and its result are worse than the two nonlinear methods GDAE and DSGNN. The result of DSGNN is better than GDAE because DSGNN emphasizes the diversity of dynamic properties of variables and considers the structure information between variables.

From Table 8, it can be found that the six methods can all successfully detect Fault 17. For the linear method TNPE, the MDR of T^2 statistic

Table 8
Missed detection rate (%) and false alarm rate (%) (shown in parentheses) of TNPE, DKPCA, DSAE, DNOM, GDAE, and DSGNN in TEP.

Fault	TNPE		DKPCA		DSAE		DNOM		GDAE		DSGNN	
	T^2	SPE	T^2	SPE	T^2	SPE	T^2	SPE	T^2	SPE	T^2	SPE
1	0.4(1.9)	0.6(2)	**0(2.5)**	**0(2.5)**	**0(1.3)**	**0(3.8)**	**0(1.9)**	**0(2.5)**	**0(0)**	0.38(0)	**0(3.8)**	**0(2.5)**
2	1.4(2.5)	2.8(0)	0.52(0.6)	3.4(2.5)	0.6(0)	1.4(0)	1(0)	0.1(11.3)	1.1(1.9)	1.5(0)	**0.5(0)**	1.1(0)
3	93.9(4.4)	99.8(0)	95.6(0)	98.9(0)	92.4(0.6)	97.5(0.6)	100(0)	89.3(2.5)	83.1(21)	93.3(0.6)	89.9(5.6)	99.1(0)
4	91.8(4.4)	100(0)	90.1(0)	98.9(0)	90.5(0)	94.2(1.9)	98(0)	0(13.1)	**0(0)**	11(0)	0.6(4.4)	40.3(1.3)
5	**0(4.4)**	0.1(0)	78.5(5.1)	78.8(5.0)	68.1(1.3)	74.5(2.5)	76.5(3.1)	56.8(13)	62(0)	72.38(0)	67.6(3.8)	74.1(2.5)
6	**0(0)**	**0(0)**	**0(0)**	1.5(0)	**0(0)**	0.4(1.3)	**0(1.3)**	**0(4.3)**	**0(0)**	0.5(0)	**0(0.6)**	**0(0)**
7	53.9(0.6)	72.5(5)	0.6(5)	0.7(0.5)	5.7(4.4)	0.9(5.6)	**0(3.8)**	0(8.8)	**0(4.4)**	0.25(3.8)	0.5(7.5)	26.5(5)
8	2.4(0)	8.3(0)	2.5(0)	3.3(0)	2.0(0.6)	2.5(0)	1.92(0)	1.5(5.6)	1.4(9.4)	2.5(0)	**1.9(1.3)**	2.6(0)
9	96.1(3.1)	99.9(1.3)	96.5(1.9)	97.9(1.9)	94(17.5)	97(2.5)	99.6(0)	89.4(7.5)	89.8(21)	98(18.8)	94.9(18)	99.8(6.3)
10	**13.4(1.3)**	77.6(0)	75(0.1)	83.3(0)	44.8(0)	78.6(0.6)	61.5(0)	49.4(2.5)	40.6(0.6)	56(0)	40.8(0.6)	61.8(0.6)
11	89.3(0)	99.9(0)	84.6(1.3)	97.5(4.4)	74.2(0)	76.5(4.4)	87.3(0)	3.9(12.5)	**27.4(2.5)**	59.5(0)	**27.4(0.6)**	62.6(0.6)
12	**0.1(2.5)**	3.4(0)	0.3(3)	22.1(0.6)	0(7.5)	0.1(6.3)	0.8(1.3)	0.1(13.1)	0.3(16.9)	1.9(1.9)	0(9.4)	0.9(0)
13	4.5(0.6)	5.5(0)	4.5(4.5)	11.8(1.4)	4.8(0)	5(1.3)	4.9(0)	4(0)	4(1.3)	9(0)	**3.9(1.9)**	4.4(0)
14	50.9(3.1)	99.9(6.9)	**0(4.4)**	**0(4.5)**	**0(3.8)**	**0(4.4)**	8(0.6)	**0(2.5)**	**0(4.4)**	1.13(1.9)	**0(4.5)**	0.1(6.9)
15	86.9(2.5)	97(0)	92.6(1)	99(3.8)	83.4(0)	93.8(1.9)	99(0)	89.5(10)	81.1(0.6)	87.63(0)	86(1.3)	95.13(0)
16	**21.8(2.5)**	92.6(5)	88(6.2)	92.5(10)	63(23.1)	88.4(8.1)	83.7(0.6)	58.1(16)	59(28.8)	74(11.9)	58(16.9)	80.6(8.1)
17	25.1(2.5)	76.9(0)	8.1(0.6)	9(1)	12(0)	15.3(2.5)	19.1(0)	2.2(5.6)	6.3(1.9)	14(0)	**5.5(0)**	13.9(0)
18	10.4(1.3)	11.1(0)	10.4(0)	13.4(0)	10.8(0)	11.1(0.6)	10.6(0)	9(9.4)	**8.1(1.9)**	10.5(0)	**8.1(1.3)**	10.5(0)
19	96.3(2.5)	99.9(0)	97.5(0)	99.5(0)	99.8(0)	97.1(1.3)	99.8(0)	15.7(3.1)	95.1(8.1)	97.6(2.5)	98(0)	99.9(0)
20	**20.9(1.3)**	49.3(0)	46.9(1)	80(0.5)	48.9(0)	60.6(0)	56(0)	26.8(5)	41.3(0.6)	69.4(0)	32.4(0.6)	63.25(0)
21	48(5.6)	46.8(0)	58.1(2.2)	74.1(0.5)	70.2(2.5)	77.9(2.5)	62.4(0)	53(6.3)	**46.6(3.1)**	73.88(0)	56.8(4.4)	73.63(0)

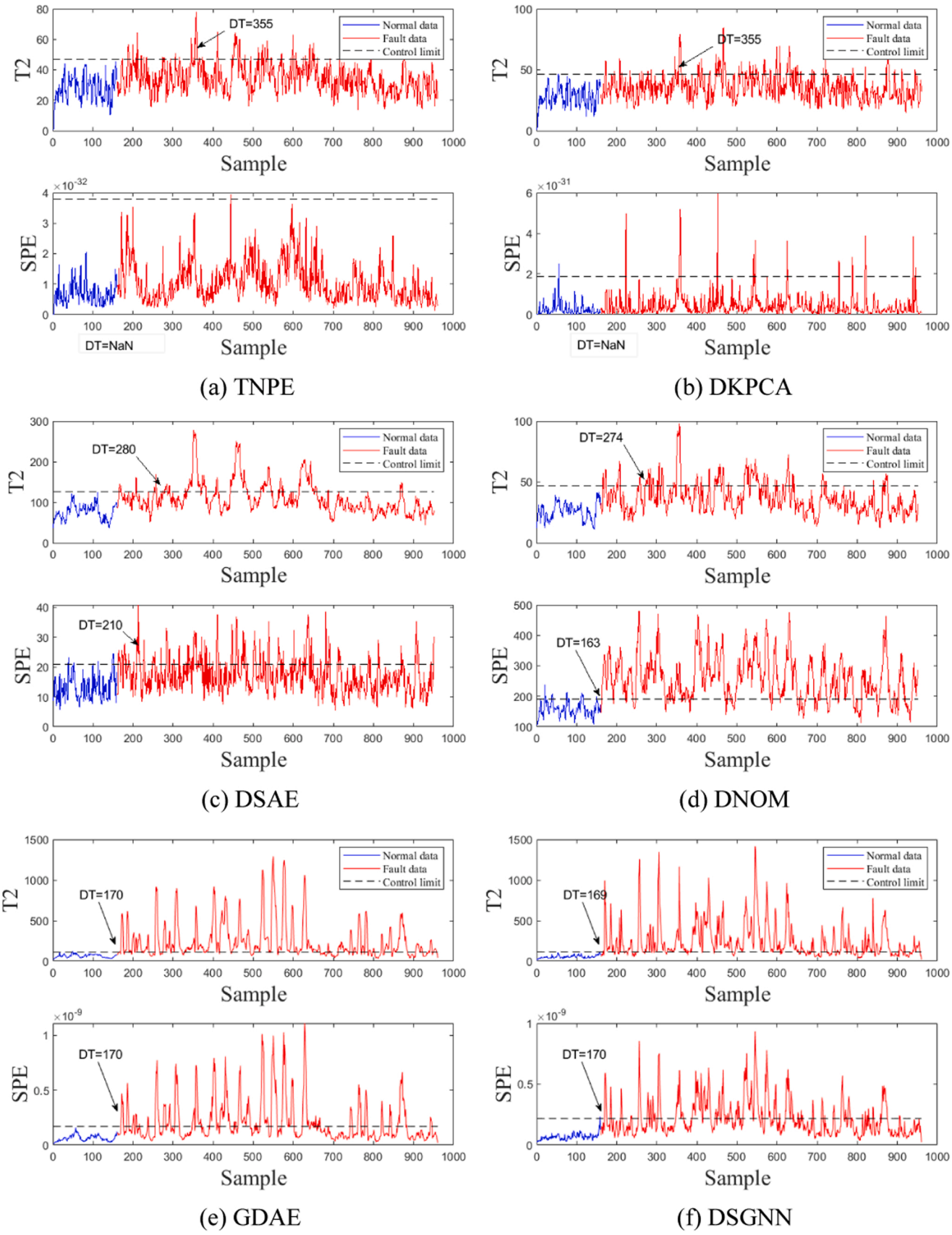

Fig. 11. Detection results of 6 different methods for Fault 11.

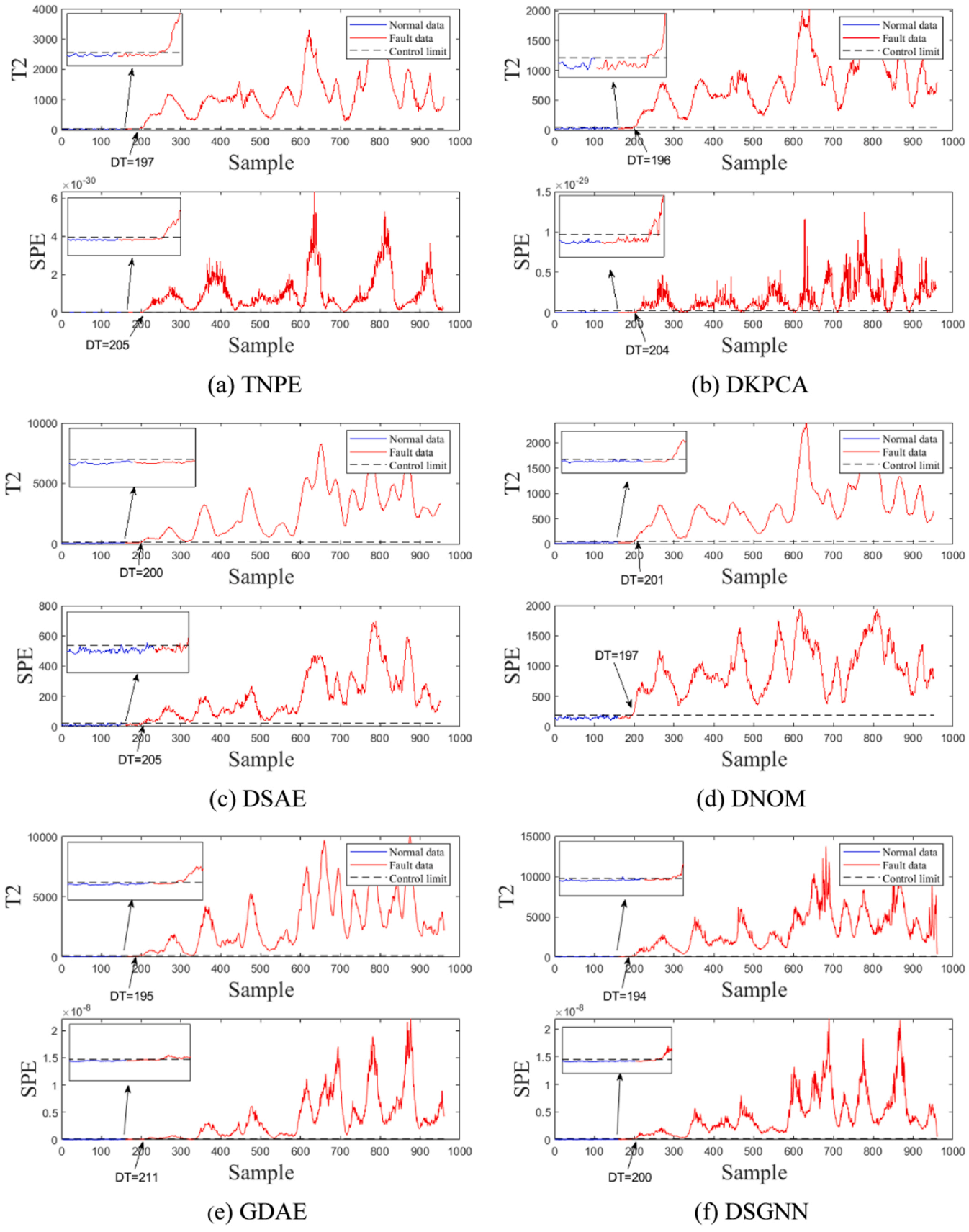

Fig. 12. Detection results of 6 different methods for Fault 13.

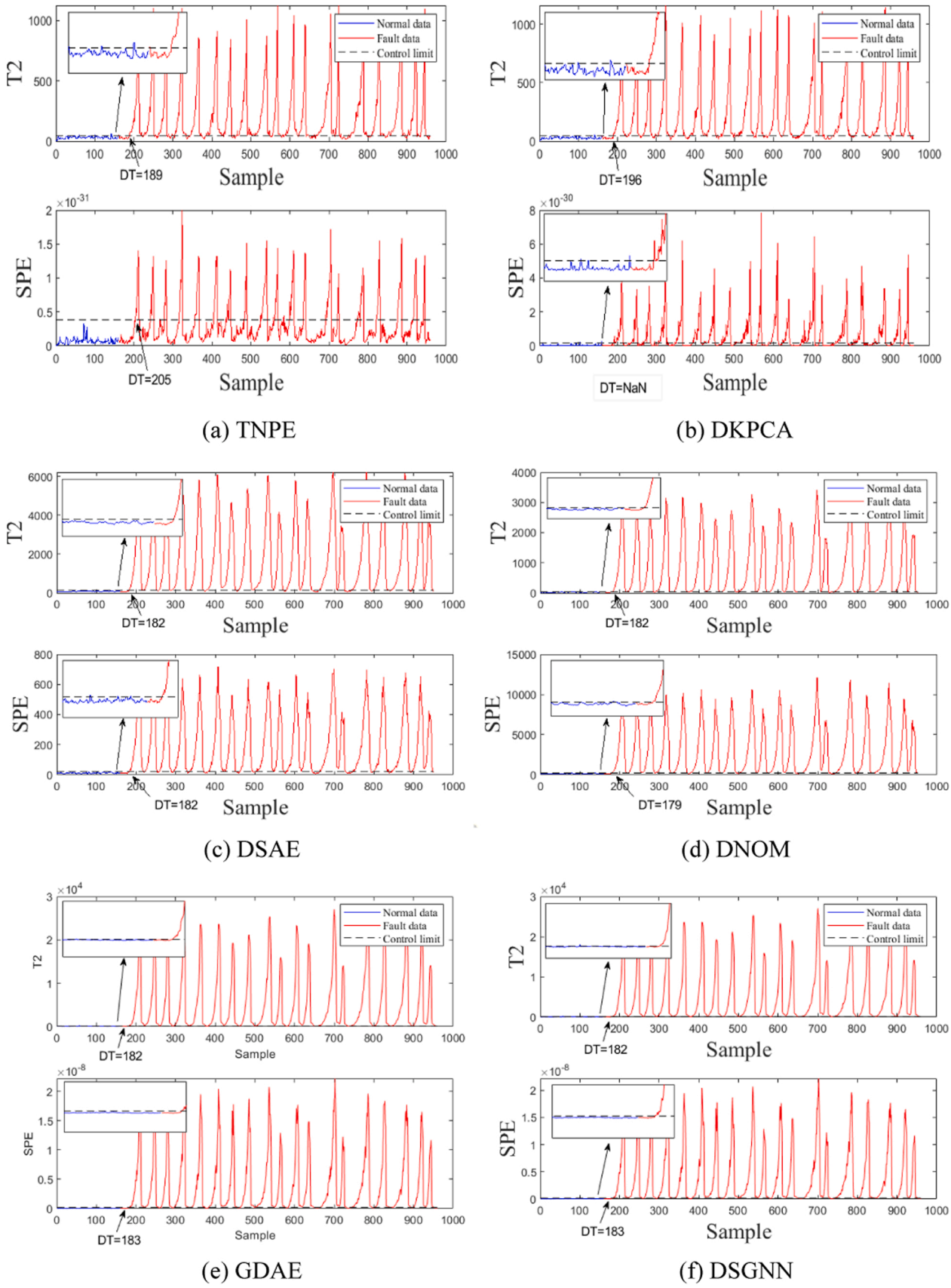

Fig. 13. Detection results of 6 different methods for Fault 17.

is 25.13% with FAR < 5%. For nonlinear methods, the MDRs of T^2 statistic of DKPCA, DSAE, DNOM, GDAE, and DSGNN are respectively 8.13%, 12.01%, 19.07%, 6.25%, and 5.5%. Fig. 13 plots the results of the six methods on Fault 17 and the small overlay plot is a partial magnification of the results near the 160*th* sample. It can be found that DKPCA, DSAE, DNOM, GDAE, and DSGNN have much better performance than TNPE, which indicates that the nonlinear method is more suitable to detect Fault 17. DSGNN combines the strength of graph-based strategy with the neural network structure, emphasizes the diverse dynamic properties of variables, and obtains better results compared with the other four nonlinear methods.

5.3. Ablation study

To evaluate the necessity of grouping according to autocorrelation and constructing dynamic-scale graphs, we conduct ablation experiments.

5.3.1. The necessity of grouping according to autocorrelation

To prove the necessity of grouping variables according to autocorrelation, we conduct ablation experiments. We design another dynamic neural network structure (DSGNN-Unit) inspired by (Jiang et al., 2021). In DSGNN-Unit, variables are grouped according to the operation units in TEP. The entire process can be divided into five operation units: a reactor, a compressor, a separator, a stripper, and an input unit. The grouping results are shown in Table 9. DSGNN-Unit constructs a subgraph in each group and the variables in the same group are considered as a subsample. The dimensionality of the subsample is the number of these variables. The number of neighbors in each subgraph is determined by the dynamic property of the sum of the variables in each graph.

Table 9
The grouping results in DSGNN-Unit.

Unit	Marker	Variable	Autocorrelation of the subsample
Input feed	1	A	1
	2	D	
	3	E	
	4	A and C	
	42	D flow	
	43	E flow	
	44	A flow	
	45	A and C flow	
Reactor	6	Feed rate	15
	7	Pressure	
	8	Level	
	9	Temperature	
	21	Water temperature	
	51	Cooling water flow	
	52	Condenser cooling water flow	
Separator	11	Temperature	15
	12	Level	
	13	Pressure	
	14	Underflow	
	22	Water temperature	
	48	Pot liquid flow	
Stripper	15	Level	15
	16	Pressure	
	17	Underflow	
	18	Water temperature	
	19	Stream flow	
	49	Liquid product flow	
	50	Stream valve	
Compressor	5	Recycle	15
	10	Purge rate	
	20	Compressor work	
	46	Compressor recycle valve	
	47	Purge valve	
The other	23–28	A, B, C, D, E, F Feed	4
	29–36	A, B, C, D, E, F, G, H Purge	
	37–41	D, E, F, G, H Product	

It should be noted that in (Jiang et al., 2021), only 11 manipulated variables and 22 measured variables are considered. For a fair comparison, we divide the component variables in TEP into the 6*th* group. The detection results are shown in Tables 10 and 11. DSGNN is considered a baseline. The up arrows and down arrows indicate higher and lower MDRs than DSGNN, and the horizontal lines indicate the same results.

5.3.2. The necessity of constructing dynamic-scale graphs

We design three dynamic neural network structures (DSGNN-OG-S, DSGNN-OG-M, and DSGNN-OG-L), each structure constructs only one graph, but the scales are different. These three structures consider the dynamic autocorrelation of the whole measured variables and ignore the dynamic autocorrelation of each variable. We design the scales of the neighborhood graph in DSGNN-OG-S, DSGNN-OG-M, and DSGNN-OG-L respectively 5, 10, and 15. We design the weight assigned to each neighbor in the three structures as the Euclidean distance normalized by softmax between the current data and the neighbor. Then we compare them with DSGNN. DSGNN is considered a baseline. The up arrows and down arrows indicate higher and lower MDRs than DSGNN, and the horizontal lines mean the same results.

DSGNN-OG-S obtains a lower MDR in only 6 cases, indicating that the structure with only 5 neighbors cannot characterize the dynamic information sufficiently in TEP. DSGNN-OG-M denotes the dynamic fault detection structure with 10 neighbors. The results of DSGNN-OG-M are better than the results of DSGNN-OG-S but worse than the results of DSGNN-OG-L and DSGNN. Although DSGNN-OG-L considers the current data with the longest time lags, the results of DSGNN-OG-L are a little worse than the results of DSGNN. It illustrates that DSGNN takes the local structure information between variables into account and obtains better monitoring performance.

Table 10
Missed detection rate (%) of DSGNN and DSGNN-Unit in TEP.

Fault NO.	DSGNN-Unit		DSGNN	
	T^2	SPE	T^2	SPE
1	0(-)	0.75(↑)	0	0
2	1.38(↑)	1.38(↑)	0.5	1.13
3	96(↑)	98(↓)	89.88	99.13
4	41(↑)	95.38(↑)	0.63	40.25
5	74.13(↑)	77.13(↑)	67.63	74.13
6	0(-)	0(-)	0	0
7	0(↓)	0(↓)	0.5	26.5
8	2.63(↑)	3.38(↑)	1.88	2.63
9	97.5(↑)	99.25(↓)	94.88	99.75
10	62.88(↑)	73.5(↑)	40.75	61.75
11	71.63(↑)	91(↑)	27.38	62.63
12	2(↑)	5.88(↑)	0	0.88
13	5.75(↑)	9.63(↑)	3.88	4.38
14	1.25(↑)	24.5(↑)	0	0.13
15	94.75(↑)	95.75(↑)	86	95.13
16	80.63(↑)	86.38(↑)	57.75	80.63
17	30.5(↑)	34.13(↑)	5.5	13.88
18	10.25(↑)	11.25(↑)	8.13	10.5
19	98.75(↑)	100(↑)	98	99.88
20	66.13(↑)	86.13(↑)	32.38	63.25
21	66.88(↑)	84.38(↑)	56.75	73.63

Compared with the baseline (DSGNN), DSGNN-Unit obtains a lower MDR than DSGNN in 4 cases of T^2 statistic and *SPE* statistic, a higher MDR than DSGNN in 35 cases, and the same MDR as DSAE in 3 cases. The performance of DSGNN-Unit is worse than DSGNN, which means grouping variables according to the operation units does not fully capture the structure information between variables.

Table 11
Missed detection rate (%) of DSGNN, DSGNN-OG-S, DSGNN-OG-M, and DSGNN- OG-L in TEP.

Fault NO.	DSGNN-OG-S		DSGNN-OG-M		DSGNN- OG-L		DSGNN	
	T^2	*SPE*	T^2	*SPE*	T^2	*SPE*	T^2	*SPE*
1	0.25(↑)	0.38(↑)	0.25(↑)	0.5(↑)	0(-)	0.13(↑)	0	0
2	1.25(↑)	1.75(↑)	1.5(↑)	1.75(↑)	1.25(↑)	1.25(↑)	0.5	1.13
3	92.63(↑)	98.88(↓)	89.25(↓)	97.38(↓)	87.38(↓)	94(↓)	89.88	99.13
4	0(↓)	50(↑)	0(↓)	23.25(↓)	0.65(↑)	41.88(↑)	0.63	40.25
5	69.25(↑)	78(↑)	58.88(↓)	75.88(↑)	62.5(↓)	75.88(↑)	67.63	74.13
6	0.25(↑)	0.63(↑)	0.5(↑)	1(↑)	0(-)	0.38(↑)	0	0
7	0(↓)	0.25(↓)	0(↓)	0(↓)	0(↓)	0(↓)	0.5	26.5
8	2.75(↑)	3(↑)	2.25(↑)	2.38(↓)	2.5(↑)	3(↑)	1.88	2.63
9	96.13(↑)	99.5(↓)	92.5(↓)	99.63(↓)	90.88(↓)	97.75(↓)	94.88	99.75
10	48.38(↑)	72.25(↑)	41.88(↑)	63(↑)	41(↑)	60.13(↓)	40.75	61.75
11	39.75(↑)	66(↑)	32.88(↑)	64.25(↑)	29.5(↑)	55.75(↓)	27.38	62.63
12	1.13(↑)	4(↑)	0.25(↑)	3.38(↑)	1.25(↑)	2.13(↑)	0	0.88
13	5(↑)	9.13(↑)	4.25(↑)	7(↑)	4(↑)	7.38(↑)	3.88	4.38
14	0.38(↑)	8.25(↑)	0(-)	1.38(↑)	0(-)	0.25(↑)	0	0.13
15	87.5(↑)	96.5(↑)	87.75(↑)	94.13(↓)	87.25(↑)	90.63(↓)	86	95.13
16	67.25(↑)	84.88(↑)	55.25(↓)	80.75(↑)	49.13(↓)	73.88(↓)	57.75	80.63
17	11.88(↑)	22(↑)	6.63(↑)	12.38(↓)	3.75(↓)	15.13(↑)	5.5	13.88
18	10.5(↑)	12.38(↑)	9.75(↑)	11(↑)	7.38(↓)	10.63(↑)	8.13	10.5
19	97.5(↓)	99.88(-)	95.88(↓)	99.75(↓)	90.25(↓)	98.75(↓)	98	99.88
20	49.25(↑)	78.75(↑)	54.63(↑)	92(↑)	40.5(↑)	82.88(↑)	32.38	63.25
21	64.25(↑)	78(↑)	59.5(↑)	74.25(↑)	55.5(↓)	82.88(↑)	56.75	73.63

Compared with the baseline (DSGNN), DSGNN-OG-S obtains a lower MDR than DSGNN in 6 cases of T^2 statistic and *SPE* statistic, a higher MDR than DSGNN in 35 cases, and the same MDR as DSAE in 1 case. DSGNN-OG-M obtains a lower MDR than DSGNN in 15 cases, a higher MDR than DSGNN in 26 cases, and the same MDR as DSAE in 1 case. DSGNN-OG-L obtains a lower MDR than DSGNN in 17 cases, a higher MDR than DSGNN in 22 cases, and the same MDR as DSAE in 3 cases. It can be found that the results are better as the number of neighbors in the graph increases.

6. Conclusion

Process monitoring and risk assessment are essential for the process industries. In this paper, we propose a novel dynamic fault detection method named DSGNN. In most dynamic process detection models, dynamic autocorrelation of the whole measured variables is considered, but dynamic autocorrelation of each variable is not analyzed. In DSGNN, the dynamic diversity of variables is fully taken into account and preserved by constructing dynamic-scale subgraphs. In each subgraph, to emphasize the different influences of neighbors on the current data, different weights are designed and assigned to different neighbors. The updated features containing the dynamic information are used as the input of the neural network and the low-dimensional global features are used for fault detection. Experimental results illustrate that our method significantly improves the monitoring accuracy and reduces the false alarm rate, so as to ensure the safe and stable operation of the process. But the proposed DSGNN framework only focuses on fault detection, ignoring the location of the cause of the fault. In future research, the DSGNN framework will be expanded to locate the cause of faults.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by National Natural Science Foundation of China (NSFC) under Grant 62173143 and 61973122.

References

Amin, M.T., Khan, F., Ahmed, S., Imtiaz, S., 2021. A data-driven Bayesian network learning method for process fault diagnosis. Process Saf. Environ. Prot. 150, 110–122.
Arunthavanathan, R., Khan, F., Ahmed, S., Imtiaz, S., 2021. An analysis of process fault diagnosis methods from safety perspectives. Comput. Chem. Eng. 145, 107197.
Choi, S.W., Lee, I.B., 2004. Nonlinear dynamic process monitoring based on dynamic kernel PCA. Chem. Eng. Sci. 59 (24), 5897–5908.
Deng, X., Tian, X., 2013. Sparse kernel locality preserving projection and its application in nonlinear process fault detection. Chin. J. Chem. Eng. 21 (2), 163–170.
Downs, J.J., Vogel, E.F., 1993. A plant-wide industrial process control problem. Comput. Chem. Eng. 17 (3), 245–255.
Fazai, R., Mansouri, M., Abodayeh, K., Nounou, H., Nounou, M., 2019. Online reduced kernel PLS combined with GLRT for fault detection in chemical systems. Process Saf. Environ. Prot. 128, 228–243.
Han, Y., Song, G., Liu, F., Geng, Z., Ma, B., Xu, W., 2022. Fault monitoring using novel adaptive kernel principal component analysis integrating grey relational analysis. Process Saf. Environ. Prot. 157, 397–410.
Harrou, F., Kadri, F., Khadraoui, S., Sun, Y., 2016. Ozone measurements monitoring using data-based approach. Process Saf. Environ. Prot. 100, 220–231.
Hu, Z., Peng, J., Zhao, H., 2021. Dynamic neural orthogonal mapping for fault detection. Int J. Mach. Learn Cyber 12 (5), 1501–1516.
Jiang, L., Ge, Z., Song, Z., 2017. Semi-supervised fault classification based on dynamic sparse stacked auto-encoders model. Chemom. Intell. Lab Syst. 72–83.
Jiang, Q., Yan, X., Huang, B., 2019. Review and perspectives of data-driven distributed monitoring for industrial plant-wide processes. Ind. Eng. Chem. Res 58 (29), 12899–12912.
Jiang, Q., Chen, S., Yan, X., Kano, M., Huang, B., 2021. Data-driven communication efficient distributed monitoring for multiunit industrial plant-wide processes. IEEE Trans. Autom. Sci. Eng. 1913–1923.
Johnson, R.A., Wichern, D.W., et al., 2014. Applied Multivariate Statistical Analysis, Vol. 6. Pearson,, London, UK.
Khan, F., Rathnayaka, S., Ahmed, S., 2015. Methods and models in process safety and risk management: Past, present and future. Process Saf. Environ. Prot. 98, 116–147.
Kingma, D.P., Ba, J., 2015. 3rd international conference on learning representations, ICLR 2015-conference track proceedings. Int. Conf. Learn. Represent., ICLR).
Kopbayev, A., Khan, F., Yang, M., Halim, S.Z., 2022. Gas leakage detection using spatial and temporal neural network model. Process Saf. Environ. Prot. 160, 968–975.
Ku, W., Storer, R.H., Georgakis, C., 1995. Disturbance detection and isolation by dynamic principal component analysis. Chemom. Intell. Lab Syst. 30 (1), 179–196.
Li, C., Mo, L., Yan, R., 2020. Rolling bearing fault diagnosis based on horizontal visibility graph and graph neural networks. 2020 Int. Conf. Sens., Meas. Data Anal. era Artif. Intell. (ICSMD), IEEE 275–279.
Li, S., Zhou, X., Pan, F., Shi, H., Li, K., Wang, Z., 2017. Correlated and weakly correlated fault detection based on variable division and ICA. Comput. Ind. Eng. 112, 320–335.
Li, Z., Yan, X., 2019. Fault-relevant optimal ensemble ICA model for non-gaussian process monitoring. IEEE Trans. Control Syst. Technol. 28 (6), 2581–2590.
Liu, L., Zhao, H., Hu, Z., 2022. Graph dynamic autoencoder for fault detection. Chem. Eng. Sci. 254, 117637.
Luo, L., Bao, S., Gao, Z., Yuan, J., 2014. Tensor global-local preserving projections for batch process monitoring. Ind. Eng. Chem. Res 53 (24), 10166–10176.
Miao, A., Ge, Z., Song, Z., Zhou, L., 2013. Time neighborhood preserving embedding model and its application for fault detection. Ind. Eng. Chem. Res 52 (38), 13717–13729.

Rong, G., Liu, S., Shao, J., 2012. Dynamic fault diagnosis using extended matrix and tensor locality preserving discriminant analysis. Chemom. Intell. Lab Syst. 116, 41–46.
Russell, E.L., Chiang, L.H., Braatz, R.D., 2000. Fault detection in industrial processes using canonical variate analysis and dynamic principal component analysis. Chemom. Intell. Lab. Syst. 51 (1), 81–93.
Samuel, R.T., Cao, Y., 2016. Nonlinear process fault detection and identification using kernel PCA and kernel density estimation, Syst Sci. Control Eng. 4 (1), 165–174.
Soltanalian, M., Stoica, P., 2012. Computational design of sequences with good correlation properties. IEEE Trans. Signal Process 60 (5), 2180–2193.
Song, B., Ma, Y., Shi, H., 2014. Multimode process monitoring using improved dynamic neighborhood preserving embedding. Chemom. Intell. Lab Syst. 135, 17–30.
Tahir, F., Islam, M.T., Mack, J., Robertson, J., Lovett, D., 2019. Process monitoring and fault detection on a hot-melt extrusion process using in-line Raman spectroscopy and a hybrid soft sensor. Comput. Chem. Eng. 125, 400–414.
Tong, H., Qiu, R.C., Zhang, D., Yang, H., Ding, Q., Shi, X., 2021. Detection and classification of transmission line transient faults based on graph convolutional neural network, CSEE. J. Power Energy Syst. 7 (3), 456–471.
Venkatasubramanian, V., Rengaswamy, R., Yin, K., Kavuri, S.N., 2003. A review of process fault detection and diagnosis: Part I: Quantitative model-based methods. Comput. Chem. Eng. 27 (3), 293–311.
Yang, J., Zhang, M., Shi, H., Tan, S., 2017. Dynamic learning on the manifold with constrained time information and its application for dynamic process monitoring. Chemom. Intell. Lab Syst. 167, 179–189.
Yao, Y., Dai, Y., Zhao, J., 2022. An enhanced dynamic artificial immune system based on simulated vaccine for early fault diagnosis with limited data. Process Saf. Environ. Prot. 165, 908–919.
Yin, J., Yan, X., 2019. Mutual information–dynamic stacked sparse autoencoders for fault detection. Ind. Eng. Chem. Res 58 (47), 21614–21624.
Yin, Q., Yang, J., Tyagi, M., Zhou, X., Hou, X., Cao, B., 2021. Field data analysis and risk assessment of gas kick during industrial deepwater drilling process based on supervised learning algorithm. Process Saf. Environ. Prot. 146, 312–328.
Zhang, Y., Fu, Y., Wang, Z., Feng, L., 2017. Fault detection based on modified kernel semi-supervised locally linear embedding. IEEE Access 6, 479–487.
Zhao, H., 2018. Dynamic graph embedding for fault detection. Comput. Chem. Eng. 117, 359–371.
Zhao, H., Yuen, P.C., Kwok, J.T., 2006. A novel incremental principal component analysis and its application for face recognition, IEEE Trans Syst Man. Cyber Syst. 36 (4), 873–886.
Zhu, J., Shi, H., Song, B., Tan, S., Tao, Y., 2020. Deep neural network based recursive feature learning for nonlinear dynamic process monitoring. Can. J. Chem. Eng. 98 (4), 919–933.