Contents lists available at ScienceDirect

Chemical Engineering Research and Design

journal homepage: www.elsevier.com/locate/cherd

A batch-wise LSTM-encoder decoder network for batch process monitoring

*Jiayang Ren, Dong Ni**

College of Control Science and Engineering, Zhejiang University, Hangzhou 310027, China

ARTICLE INFO

Article history:
Received 1 May 2020
Received in revised form 10 September 2020
Accepted 14 September 2020
Available online 24 September 2020

Keywords:
Nonlinear batch processes
Process monitoring
Multi-layer LSTM
Encoder–decoder structure
Kernel density estimation

ABSTRACT

Process monitoring is essential to keep quality consistency and operation safety in the batch process. However, the existence of multiphase, nonlinearity and dynamic features in the batch process makes the batch process monitoring a complicated task. In this work, a multi-layer recurrent neural network in the encoder–decoder structure called batch-wise LSTM-encoder decoder network is proposed to solve the difficulties mentioned above in batch process monitoring. The LSTM-encoder extracts the nonlinear dynamic features in both between and within batch direction, then projects the high dimensional input space to a low dimensional hidden state space. The decoder part regenerates the samples from hidden states. Control statistics H^2 and *SPE* are designed for process monitoring, and the corresponding control limits are estimated by kernel density estimation. A case study on an extensive reference penicillin fermentation dataset suggests that the proposed method can detect the fault samples more effectively than previous methods while keeping the same robustness in normal conditions.

1. Introduction

Batch processes play a significant role in modern industries such as pharmaceutical, semiconductor and biotechnology (Wang et al., 2019). It is defined as a type of process that repeatably produces a finite quantity of products once a time following a sequence of process steps (Mehta and Reddy, 2015). Because of the flexibility in production, it is important to keep quality consistency and operation safety in the batch process. Process monitoring, or in other words, online fault detection, is vital to meet the demand of quality and safety in the batch process, so it has become a hot field in the past decades (Qin, 2012).

Owing to the abundant sensors linking to the network in the modern batch process, statistical process control (SPC) methods have been widely applied to monitor and control the process (Yao and Gao, 2009). Univariate control chart is a feasible and popular method to monitor the process. However, it suffers from "blind spots" in a multivariate setting, which would loosen the control limits (Wang et al., 2017). In order to overcome these difficulties, many multivariate process monitoring methods have been developed. Compared to the continuous processes, the dynamic features within and between batches make the monitoring of the batch process more complex and challenging. In all the methods suitable for batch processes, multiway principal component analysis (MPCA) is the most widely used one (Nomikos and MacGregor, 1994). The multiway methods firstly unfold the three-dimensional (*Batch* × *Time* × *Variable*) historical batch process data cube to a two-dimensional matrix, such as batch-wise unfolding, *Batch* × (*Time* × *Variable*) and variable-wise unfolding, *Variable* × (*Time* × *Batch*). Then, the PCA method is applied to these two-dimensional matrices to project the high-dimensional data into a low-dimensional space and the statistics such as Hotelling's T^2 and square prediction error(*SPE*) are adopted to measure the main space and residual information.

However, MPCA methods are typical linear methods which may fail to extract the nonlinear information in the batch process. In order to improve the nonlinear process monitoring efficiency, many modified methods based on MPCA are proposed in the past decades. Among them, multiphase methods (Lu et al., 2004; Zhao and Sun, 2013) and kernel methods (Lee et al., 2004; Jia et al., 2010) are two mainstream methods. Mul-

* *Corresponding author.*
E-mail address: dni@zju.edu.cn (D. Ni).
https://doi.org/10.1016/j.cherd.2020.09.019

tiphase methods are similar to piecewise linearization. They first divide the batch into several phases containing relatively similar dynamic characteristics. Then local MPCA models are built for each phase separately. The ability of extracting nonlinear information is improved by reducing nonlinearity in one phase (Lu et al., 2004). Kernel PCA uses kernel functions to map data from low-dimensional input space to high-dimensional feature space, then apply PCA in the high-dimensional feature space (Lee et al., 2004).

Nowadays, artificial neural networks (ANN) are getting more attention for the ability of effectively extracting nonlinear features (Adamowski et al., 2012). Particularly, autoencoders (AEs) and recurrent neural networks(RNN) are two widely studied networks in the field of process monitoring (Yan et al., 2016; Yu and Zhao, 2019; Shahnazari, 2020; Zhang et al., 2019; Loy-Benitez et al., 2020). AEs is a type of multilayer neural network with a low-dimensional central layer and is effective in dimension reducing and variance capturing in the nonlinear continuous processes according to (Yan et al., 2016; Yu and Zhao, 2019). In Yan et al. (2016), variant autoencoders including denoising autoencoders and contractive autoencoders are applied to the Tennessee Eastman process (TE process) and monitors the process by statistics H^2. In Yu and Zhao (2019), denoising autoencoders along with elastic net is proposed to generate a sparse and robust network. Statistics A^2, *SPE* and *C* are used to monitor the main hidden space, residual information, and overall measurement in the continuous processes. Considering the nature of time sequences, RNN is extensively studied for the ability of capturing sequential and dynamic information in the continuous process (Shahnazari, 2020). In Shahnazari (2020), a bank of RNNs is used to build the predictive model from the input sequence, and process monitoring is done by comparing the predictions and realistic samples. Moreover, the networks combining AEs and RNN are carried out in the field of continuous process monitoring to utilize the ability of dimension reducing and sequential information capturing (Zhang et al., 2019; Loy-Benitez et al., 2020). In Zhang et al. (2019), UNet based model, which is a variant of AEs with LSTM linking the corresponding encoder and decoder layer, is proposed to extract the nonlinear features along with time sequence information and is applied to monitor a power plant. In Loy-Benitez et al. (2020), a multi-input multi-output RNN encoder called MG-RNN-AE is designed to reduce dimension and reconstruct the original input sequence, and process monitoring is realized by monitoring the statistics H^2 and *SPE*.

However, all the above mentioned neural networks can only extract the nonlinear feature between samples. As a characteristic of the batch process, the dynamic features not only exist between samples within a batch but also between batches. This concern indicates that some specific designs could be carried out to improve the monitoring performance in the nonlinear batch process. In Jiang et al. (2020), the three-way batch process data cube is firstly unfolded to a two-dimensional matrix. An AE model is constructed to extract the relations between variables. Finally the dynamic features between batches are captured by canonical correlation analysis (CCA) of 2-D matrices, including samples from different batches. In this way, the synthetic hidden state of AEs and CCA is more suitable to monitor the batch processes.

In order to address the above concerns, a batch-wise LSTM-encoder decoder network, along with the monitoring scheme for the nonlinear batch process, is proposed. The main contribution of the proposed method is that the LSTM-encoder can capture the between and within batch dynamic features in the nonlinear batch process simultaneously by sliding through the batch-wise input sequence. Specifically, the batch-wise LSTM-encoder is a multi-layer recurrent neural network with a shrinking layer dimension. The decoder part is a symmetric multi-layer neural network. The input sequence of LSTM-encoder is a series of *k*th samples from different batches. The encoder slides through these input sequences to obtain the between batch features and shrinks layer by layer to obtain the within batch features. After encoding, the last hidden state of the input sequence is passed to the decoder to reconstruct the input sample. Batch process monitoring is achieved by monitoring the proposed H^2 and *SPE* statistics. A case study of an extensive reference data set associated with the PenSim benchmark model proposed by Van Impe and Gins (2015) shows an outstanding process-monitoring performance, especially for the barely detectable faults comparing to the AEs and MPCA monitoring model.

2. Preliminary

2.1. Multi-layer RNN

Recurrent neural networks (RNN) is a kind of neural network used for variable sequences ($X = (x_1, x_2, \ldots, x_t)$) modeling. Distinguished from other neural networks, RNN consists of a hidden state *h* and a feedback loop of hidden states to update the current hidden state, which introduces the information from previous timestamps to the current state. As shown in Fig. 1, RNN could be either a single-layer or multi-layer structure. For a *J* layers RNN, the current hidden state h_t^j of *t*th time stamp and *j*th layer and the optional output o_t are updated by

$$\begin{aligned} h_t^j &= \begin{cases} f_h^1(x_t, h_{t-1}^1), & j = 1 \\ f_h^j(h_t^{j-1}, h_{t-1}^j), & 1 < j \le J \end{cases} \\ o_t &= f_o(h_t^j) \end{aligned} \tag{1}$$

where f_h and f_o is the hidden and output unit activation functions, respectively. The activation functions can be nonlinear functions like "Tanh" and "Sigmoid" function.

2.2. Long short-term memory

Long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) is a specific RNN unit that is designed to conquer the gradient vanish and long-term memory loss problem. Besides the hidden state, it has a cell state in each unit to remember values from past and three gates called the input, forget and output gate to control the accepted values from the current input, the past cell state, and the current cell state respectively. Each part of the LSTM unit is updated by

$$\begin{aligned} i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\ f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\ g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\ o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\ c_t &= f_t * c_{t-1} + i_t * g_t \\ h_t &= o_t * \tanh(c_t) \end{aligned} \tag{2}$$

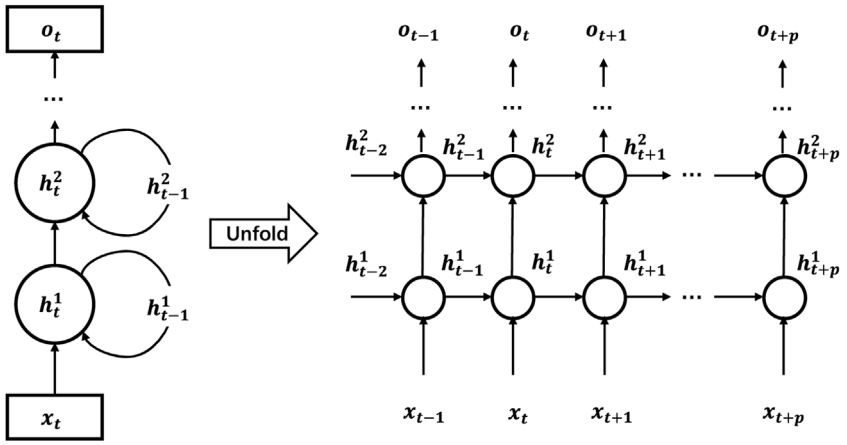

Fig. 1 – The structure of multi-layer RNN.

where h_t, c_t are the hidden state and cell state at time t respectively, i_t, f_t and o_t are the input, forget and output gate, g_t is the cell gate, σ is the sigmoid activation function, $*$ is the Hadamard product. It is needed to be clarified that we did not use peephole connections in this paper because it has a similar performance with the basic LSTM structure as compared in Breuel (2015).

3. Methodology

3.1. Batch data description

Measured process data in the multiphase batch process are usually stored in the form of a three-dimensional cube, $X \in \mathbb{R}^{I\times J\times K}$, recording K measured points with J process variables of all the I batches, as shown in Fig. 2. It should be noted that the lengths of batches are unfixed, which is not represented in Fig. 2. Nevertheless, the unfixed batch length problem will be considered in this study, and the sample rates of all the variables are considered equal in this study. Time slice, $x_k \in \mathbb{R}^{I_k\times J}$ is used as the basic unit in this study. In this way, the data cube X can be re-organized as $X = \{x_1, \ldots, x_k, \ldots, x_K\}$ where K denotes the longest trajectory length in all the batches.

The measured process data is normalized to the range of [0, 1] by the maximum and minimum value of each variable before offline model training. The new coming data sample in online monitoring is normalized by the same maximum and minimum value of each variable as the offline model training.

3.2. Batch-wise LSTM-encoder decoder

The proposed batch-wise LSTM-encoder decoder takes moving windows at batch-wise in the time slice x_k as inputs, then encodes them through the multi-layer shrinking LSTM-encoder, and finally reconstructs them in a symmetrical multi-layer BP decoder. In this way, the variance of trajectories among batches is recorded in the LSTM cell, and the dimension of variables is diminished while maintaining the main trajectories and decreasing the noise signals.

As shown in Fig. 2, we denote the sample of bth batch at time k as $x_k^b \in \mathbb{R}^J$. The input sequence of the proposed Batch-Wise LSTM-Encoder is denoted as $X_k^b = \{x_k^{b-w+1}, \ldots, x_k^b\} \in \mathbb{R}^{w\times J}$, in which w is the length of batch-wise moving window. At the initial $w-1$th batches, the input is $X_k^b = \{x_k^1, \ldots, x_k^b\} \in \mathbb{R}^{b\times J}$, $0 < b < w$. Considering the length of batches is unfixed, there will be an unaligned time slice near the ending part of the batches. For these ending time slices, the same steps as the normal time slices are taken to generate the batch-wise moving window input X_k^b except the batches are not originally continuous in these time slices.

As shown in Fig. 2, the proposed batch-wise LSTM-encoder is a multi-layer LSTM network with hidden state dimension reduction. The LSTM cell of each layer enrolls the input sequence X_k^b or the hidden state sequence from the last layer sequentially and calculates the hidden state according to Eqs. (1) and (2). After traversing the input sequence layer by layer, the bth batch hidden state of deepest layer, $h_{k,b}^{deepest}$, is summarized as the output of encoder and the input of the decoder.

The decoder of the proposed model is a symmetric fully connected network of the LSTM-encoder, which means the dimension of each layer in the decoder is the same as the corresponding LSTM layer. The decoder is trained to regenerate the bth batch sample,x_k^b, of the input sequence, $X_k^b = \{x_k^1, \ldots, x_k^b\}$, by

$$\hat{x}_k^b = f_d(h_{k,b}^{deepest}) \tag{3}$$

where f_d is the multi-layer FC network function made up of linear function layers followed by a nonlinear activation function layer ($\sigma(W\times x+b)$). The nonlinear activation functionσ can be "Sigmoid", "ReLU" (Glorot et al., 2011) or "SeLU" (Klambauer et al., 2017).

The parameter set, θ, of the batch-wise LSTM-encoder decoder can be jointly estimated by solving the following optimization problem:

$$\theta = arg\min_\theta L(x_k^b, \hat{x}_k^b), \tag{4}$$

$$L(x_k^b, \hat{x}_k^b) = \frac{1}{J}\sum_{j=1}^{J}(x_{k,j}^b - \hat{x}_{k,j}^b)^2$$

where $x_{k,j}^b$ and $\hat{x}_{k,j}^b$ are the jth variable value of the original sample and regenerated sample correspondingly. The optimization problem can be solved by a gradient-based algorithm like 'SGD' (Bottou, 2010) or 'Adam' (Kingma and Ba, 2014).

After the batch-wise LSTM-encoder decoder is trained, the monitoring model can be constructed by collecting the network outputs of the normal historical batch process data.

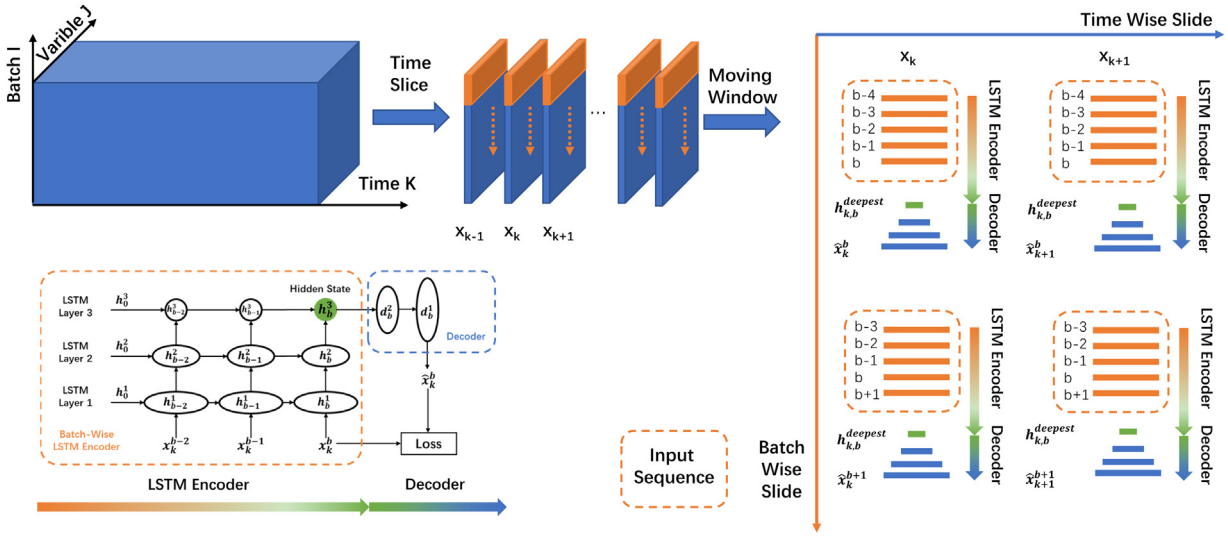

Fig. 2 – A simple illustration of the LSTM-encoder decoder network structure.

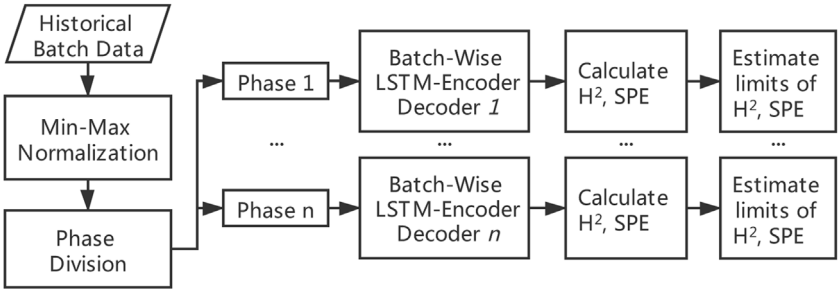

Fig. 3 – Flow chart of offline process modeling.

3.3. *Offline process modeling*

Like the popular batch process monitoring models, we try to extract the main trajectories of historical normal process data and design some statistics, like T^2 and *SPE*, to measure the deviation of the current sample from the main trajectories in the offline process modeling. Considering the nonlinearity and dynamic features among batch direction, the proposed batch-wise LSTM-encoder decoder is utilized as the main trajectory extraction in this study. Moreover, the historical batch data is divided into several phases because of the significant difference between different phases. Here, because the main concern of this study is not how to divide phases and the proposed model can catch the nonlinearity characters in the trajectories, simple division methods like indicator variable method would be enough to divide the phases.

Fig. 3 shows the flow chart of the offline process modeling. Historical normal batch process data are first normalized by the global minimum and maximum value of each variable in the dataset as Eq. (5):

$$\bar{x}_{k,j}^{b} = \frac{x_{k,j}^{b} - x_{\min,j}}{x_{\max,j} - x_{\min,j}} \tag{5}$$

where $x_{k,j}^{b}$ is the kth time, jth variable value of bth batch, $x_{min,j}$ and $x_{max,j}$ is the global minimum and maximum value of jth variable. In the following description, without a special statement, we use x to represent normalized data $\bar{x}$.

After normalization, the process is divided into several phases, and a local batch-wise LSTM-encoder decoder network is trained for each phase. In order to monitor the deviation from the main trajectories, we proposed statistics H^2 and *SPE*, similar to the statistics T^2 and *SPE* in the MPCA model (Nomikos and MacGregor, 1994). H^2 measures the deviation of the current sample from the historical average in the output hidden state of LSTM-Encoder, which can be calculated by

$$H_{k,b}^{2} = \sum_{j=1}^{R} \left(h_{k,b,j}^{deepest} - h_{k,j}^{mean}\right)^2 / h_{k,j}^{variance} \tag{6}$$

where H_k^2 is H^2 of the kth sample in bth batch, R is the dimension number of LSTM-encoder output hidden state, $h_{k,b,j}^{deepest}$ is the output hidden state value of kth sample, bth batch, and jth dimension, $h_{k,j}^{mean}$ and $h_{k,j}^{variance}$ are the average and variance value of kth sample, jth dimension output hidden state of all the training batches.

One LSTM-encoder output hidden state is corresponding to a set of inputs since the hidden state dimension is smaller than the inputs. However, one LSTM-Encoder output hidden state can only generate one hat sample after decoding. While H^2 can represent the deviation of the LSTM-encoder output, the residual of the decoder output is not measured yet. Therefore, statistics *SPE* are proposed to measure the residual error of the regenerated sample from the original sample, which can be calculated by

$$SPE_{k,b} = \sum_{j=1}^{R} \left(x_{k,j}^{b} - \hat{x}_{k,j}^{b}\right)^2 \tag{7}$$

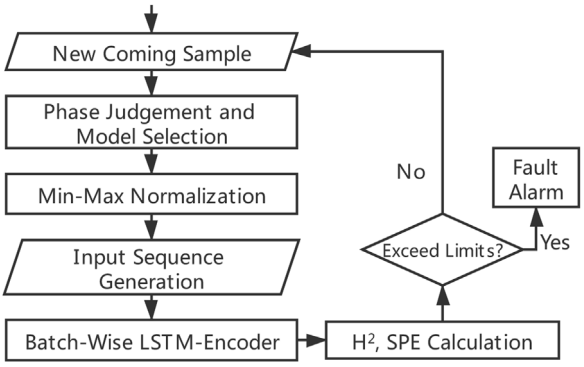

Fig. 4 – Flow chart of online process monitoring.

where $SPE_{k,b}$ is the SPE of the kth sample in bth batch, $x^b_{k,j}$ and $\hat{x}^b_{k,j}$ are the input and regenerated value of the kth sample, bth batch, and jth dimension.

In the PCA-based monitoring model, the control limits of statics T^2 and SPE can be conducted to be F distribution and weighted Chi-squared distribution under the assumption that the training data follows Gaussian distribution (Nomikos and MacGregor, 1995). However, in the proposed method, the training data are not assumed to follow Gaussian distribution. So kernel density estimation (KDE) (Parzen, 1962) is adopted to estimate the distribution of H^2 and SPE at batch-wise separately. Specifically, the kernel function in this study is the Gaussian kernels and the uni-variate data can be estimated by

$$\rho_K(y) = \sum_{i=1}^{N} K((y - x_i)/w), \tag{8}$$

$$K(x; h) \propto \exp\left(-\frac{x^2}{2h^2}\right)$$

where y is the point to be estimated, x_i is the element in H^2 or SPE set, N is the number of H^2 or SPE set, and w is the bandwidth, which determines the smoothness of the distribution. Because the ranges of h^2 and SPE both start from zero, the control limit of each time interval, $H^2_{control,k}$ and $SPE_{control,k}$, are set to the upper bound of α confidence interval starting from zero.

3.4. Online process monitoring

For online process monitoring, the goal is to judge whether the control statistics of the new coming sample exceeding the control limits. The point that exceeds the limits is rarely existing in the normal historical data, which means it could be a fault. Considering fault batches are the minority in the online monitoring, one fault batch mostly follows several normal batches, which may diminish the fault magnitude in the batch-wise LSTM loop. So in the online process monitoring, the input sequence of LSTM-encoder is composed of w repeated new coming sample to enhance the fault influence.

The procedure of online process monitoring is shown in Fig. 4 and described as follow:

1. The new coming sample, $x_{new,k} \in \mathbb{R}^J$, is allocated to a specific phase according to the process time k in the batch.
2. The corresponding monitoring model is selected to work, and the sample is normalized by the corresponding global minimum and maximum of each variable in the training dataset.
3. The new coming sample is copied w times to generate the input sequence, $X_{new,k} = \{x_{new,k}, \ldots, x_{new,k}\} \in \mathbb{R}^{w \times J}$.
4. The hidden state of the new coming sample, $h^{deepest}_{new,k}$ and the regenerated sample, $\hat{x}_{new,k}$ are calculated by the trained batch-wise LSTM-encoder decoder model (Eqs. (1)–(3)).
5. The control statistics $H^2_{new,k}$ and $SPE_{new,k}$ are calculated by Eqs. (6) and (7).
6. The corresponding control limits of kth time, $H^2_{control,k}$ and $SPE_{control,k}$ are compared. Either of the statistics exceeding the control limits would lead to a fault alarm. Otherwise, recirculate to Step 1 and wait for the next new coming sample.

3.5. Discussion

3.5.1. About the network structure

In this work, the network structure includes the number of layers and the number of hidden neurons in each layer. It would influence the expressions of the dataset, which may cause over-fitting or under-fitting problem. The performance of network structure can be evaluated by MSE loss as Eq. (4). Specifically, the network structures need to be determined case by case (Nielsen et al., 2020). Firstly, the initial setup is selected manually. Then the optimization of structure can be realized by an early stopping strategy (Wang et al., 2019), in which the number of each layer's hidden neuron is set to the local optimums with lowest MSE loss.

3.5.2. About the network training process configuration

The purpose of the network training process is to minimize the loss function while preventing possible over-fitting. In this work, the results of network training are evaluated by training loss and validation loss. Accurately, we split the normal dataset into a training dataset and validation dataset. The training dataset is used to train the network, and the validation dataset is used to evaluate whether the network starts over-fitting. The training configuration includes training epochs, learning rates, optimization methods, and training batch size. Because we split several phases in one batch and build a local model for each phase, there would be lots of configuration work for only one batch process. So the principle of configuration in this work is to find a unified, well-optimized but not over-fitted combination for all the local models. Following this principle, Adam optimizer (Kingma and Ba, 2014) and epoch-wise decaying learning rate strategy are selected. The training process stops at the first epoch in which the training loss decay rates of all the local networks are smaller than a criterion, and the validation losses are not noticeable growing.

3.5.3. About the selection of control limits

The selection of control limits is a dilemma of sensitivity and robustness. Generally, a relatively low false alarm rate (like <5%) is required in the practical application. High rates of false alarms would result in a trust problem in the application. In this work, the fault detection rate and false alarm rate heavily depend on the confidence interval α in the selection of control limits. So the confidence interval α is recommended to be set to the point with an acceptable false alarm rate and a relatively high fault detection rate.

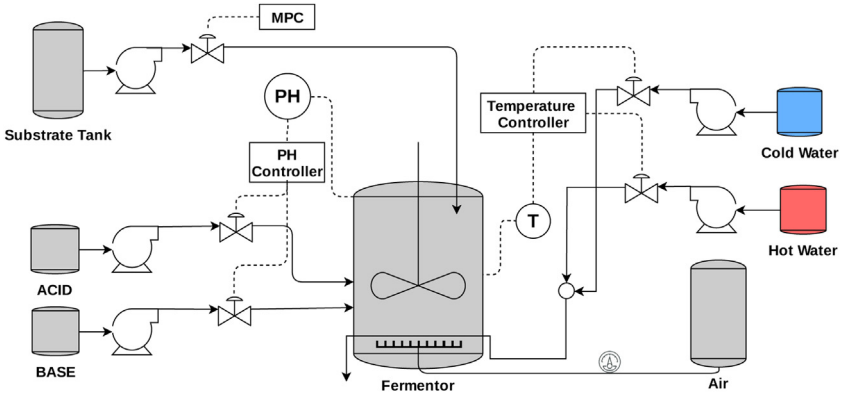

Fig. 5 – Schematic diagram of the fermentation process for penicillin production.

Table 1 – Description of different types of fault upsets.

No.	Fault &magnitudes
1	Sudden change in feed substrate concentration (%)−10, −5, −2, −1, −0.5, +0.5, +1, +2, +5, +10
2	Change in coolant temperature (°C) −2, −1, −0.5, −0.2, −0.1, +0.1, +0.2, +0.5, +1, +2
3	Agitator power drop (%)−0.5, −1, −1.5, −2, −3, −4, −5, −10
4	Aeration rate drop (%)−5, −10, −15, −20, −25, −30, −50, −70
5	Gradual dissolved oxygen sensor drift (saturating at 0.2/2 for negative/positive drifts, %/h) −0.10, −0.05, −0.02, −0.01, −0.005, +0.005, +0.01, +0.02, +0.05, +0.10
6	Feed temperature change (drift with +1.5°C to indicated level, °C) + 0.5, +1, +2, +5, +10, +20, +40, +60
7	pH sensor drift (saturating at +2, /h) + 0.001, +0.002, +0.003, +0.004, +0.005, +0.010, +0.015, +0.025
8	Non-functional pH control (no acid or base flow for indicated duration, h) 0 .5, 1, 2, 5, 10, 20
9	Reduced pH control (control action and maximal control action reduced by indicated fraction, %) −10, −20, −40, −60, −80, −90
10	Reactor temperature sensor bias (°C) −0.50, −0.10, −0.05, +0.05, +0.10, +0.50
11	Reactor temperature sensor drift (saturating at −5/+5°C for negative/positive drifts, °C) −0.10, −0.05, −0.01, −0.005, +0.005, +0.01, +0.05, +0.10
12	Reduced temperature control (control action and maximal control action reduced by indicated fraction, %) −5, −10, −20, −30, −40, −50
13	Reduced temperature control (control action reduced by indicated fraction, maximal flow not impacted, %) −10, −20, −30, −40, −50, −60
14	Contamination (drift of YP|S with −0.05/h to indicated level) −0.05, −0.10, −0.15, −0.20, −0.25

4. Case study

The PenSim benchmark model (Birol et al., 2002) is chosen as the test case in this study for its popularity in the field of batch process monitoring studies (Wang et al., 2019, 2018). It describes the production of penicillin in a fed-batch reactor, as shown in Fig. 5. Instead of using the simulation dataset created by ourselves, a public extensive reference dataset (Van Impe and Gins, 2015) based on the PenSim benchmark model is used in this study. In this reference dataset, the researchers carefully design the simulation conditions to obtain a more convincing and realistic dataset. The following factors are included in the dataset: (i) Gaussian measurement noises are added to the measured variables to match the practical situation, (ii) extensive batches are simulated to gain a statistically meaningful dataset, (iii) different process upsets including different types, magnitudes and onset times are chosen to simulate the faults conditions. These factors make the reference dataset suitable for this study.

In the reference dataset, each sample contains 11 variables. A varying number of samples around 2300 are measured in each batch. Two hundred normal batches are used to train the proposed monitoring model, and fourteen types of faults with extensive batches are tested to compare the efficiency of the proposed monitoring model with other popular models like SubPCA (Lu et al., 2004) and AEs. The specific types and fault magnitudes of different fault batches can be found in Table 1. For each magnitude of fault in each type, one hundred batches with different onset times are collected.

4.1. *Process modeling*

In the reference dataset, there are a total of 200 normal batches in which the first 150 batches are chosen as the training

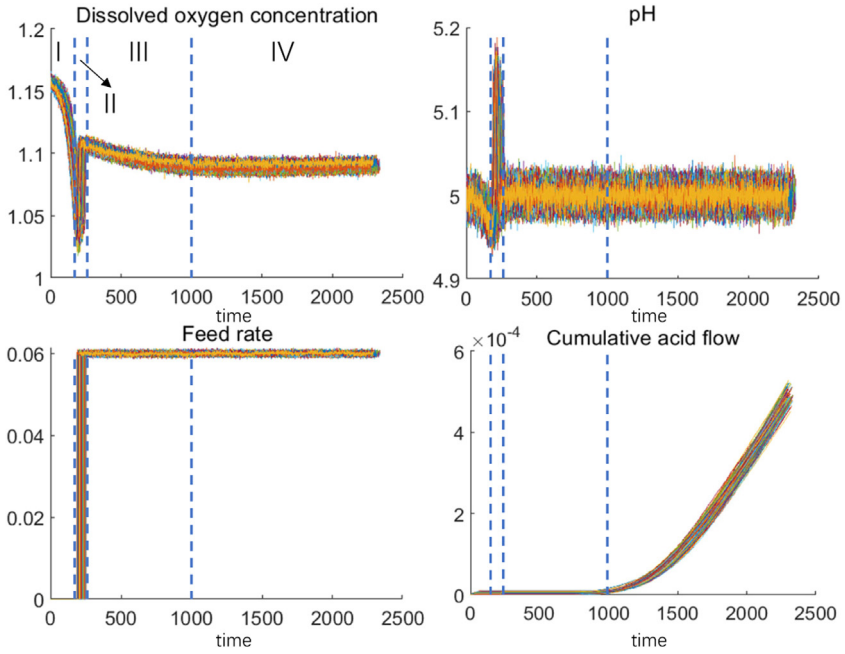

Fig. 6 – Indicator variables in the normal dataset.

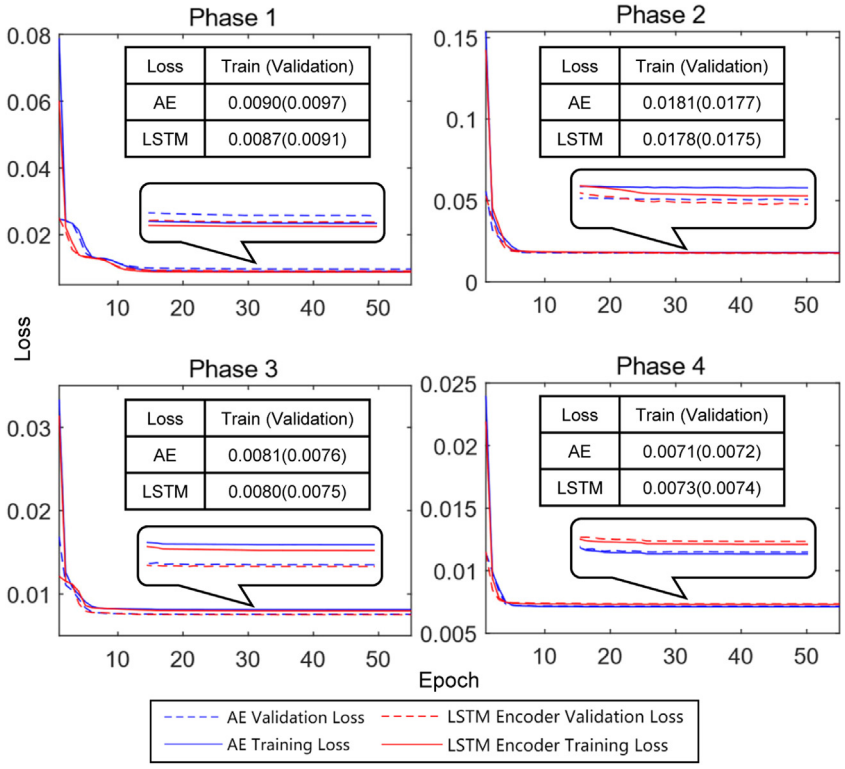

Fig. 7 – Averaged training and validation loss for each sample of AEs and the proposed LSTM-encoder.

dataset, and the remaining 50 batches are used to validate the monitoring results in the normal conditions. Fig. 6 shows four variable trajectories of the training batches. These variables are piecewise stationary or piecewise monotony, so they are chosen as the indicator variables to divide phases roughly. It can be seen that these indicator variables switch states at around time k = 180, 300 and 1000. Hence, four phases are divided: 1–179, 180–300, 301–999, and 1000–the end of each batch. Before local model training, the data of each phase are normalized separately by the maximum and minimum values in each phase.

For each phase, local models, including the proposed batch-wise LSTM-encoder decoder, AEs, and variable-wise unfolding MPCA (Sub-PCA) models, are built. Specifically, the numbers of neurons in the hidden layers of the LSTM-Encoder and AEs are both set to 11-8-4-2-4-8-11. ReLU function is selected as the activation function. The batch-wise window size w in LSTM-Encoder is set to 5. Mean square error (MSE) function (as shown in Eq. (4)) is selected as the loss function to optimize. Adam optimization method (Kingma and Ba, 2014) is used to solve the optimization problem, and the training batch size is set to 128 samples. The learning rates of these two neural networks

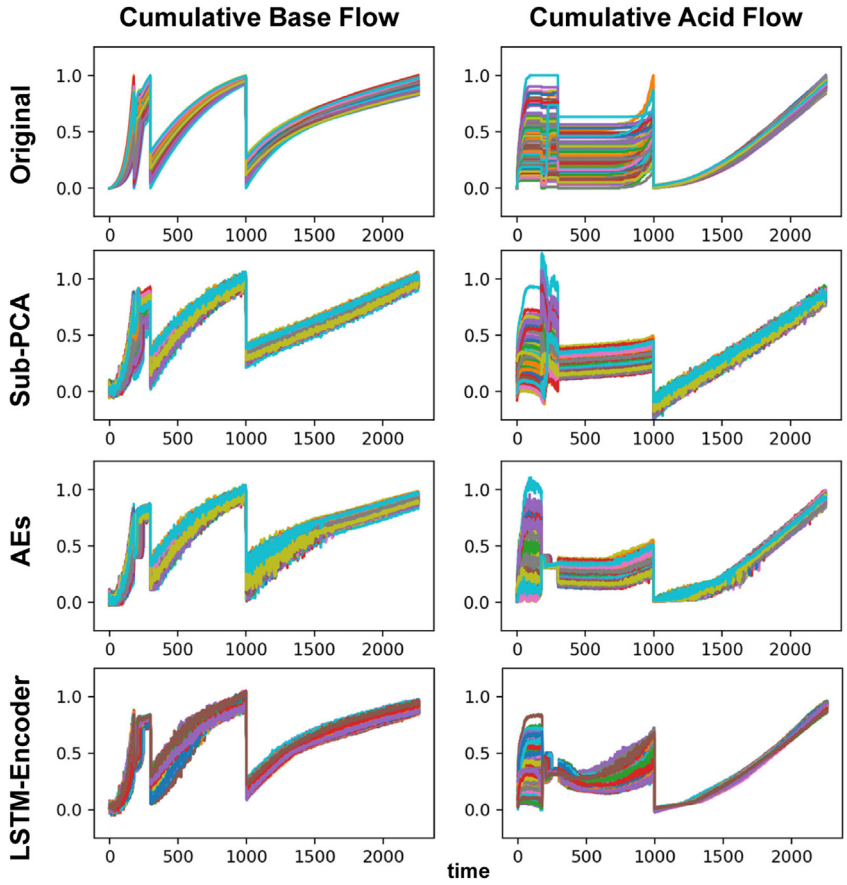

Fig. 8 – Comparison of original and regenerated variables.

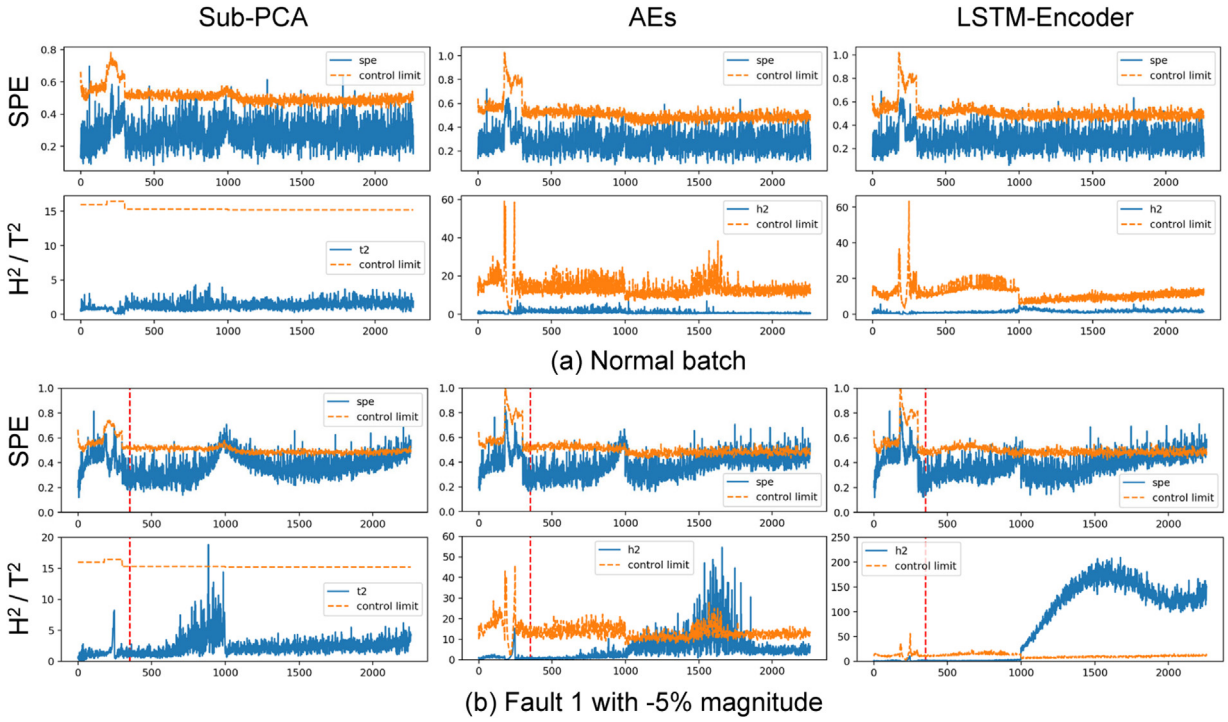

Fig. 9 – Monitoring result comparison of normal batch (a) and fault batch (b).

are set to 0.001 with decaying at the rate of 0.1 every 15 epochs, and 55 epochs are trained to reach the stable loss in all the local models.

Fig. 7 shows the training loss and validation loss of the local models during the training procedure. The training losses and validation losses are all stable with minor changes and no sign of rising after 55 training epochs, which indicates the models have reached the optimum set. In phase 1, 2 and 3, the losses of the proposed LSTM-encoder decoder models are smaller than AE models while in phase 4, a little bigger. Although this is not direct evidence that the proposed method can perform better in the process modeling, it indicates the proposed method can

Table 2 – Averaged FDRs for each type of fault with different magnitudes under 99% confidence interval (format: Sub-PCA/AEs/LSTM-encoder, units: %, bold values: fault types in which LSTM-encoder performs better, first row: number of fault magnitude).

Magnitude number	1	2	3	4	5	6	7	8	9	10
Fault 1	**56/58/65**	**16/22/51**	**2/2/26**	**1/1/11**	**1/1/6**	**1/1/3**	**1/1/9**	**2/2/26**	**17/20/50**	**67/69/76**
Fault 2	**86/85/87**	**25/25/30**	**4/4/8**	**1/2/3**	**2/2/5**	**1/1/2**	**2/2/4**	**6/6/10**	**64/67/71**	93/94/93
Fault 3	**2/2/4**	**7/7/10**	**25/25/27**	58/57/58	98/97/97	100/100/100	100/100/100	100/100/100	–/–/–	–/–/–
Fault 4	**1/1/3**	**2/2/4**	**6/6/7**	**17/17/18**	46/46/46	**77/77/78**	99/99/99	99/99/99	–/–/–	–/–/–
Fault 5	**93/93/94**	88/88/88	**70/71/71**	**52/52/53**	25/25/25	**25/25/26**	**50/50/52**	**72/72/73**	**88/88/89**	93/93/93
Fault 6	100/100/100	100/100/100	100/100/100	100/100/100	100/100/100	100/100/100	100/100/100	100/100/100	–/–/–	–/–/–
Fault 7	**10/13/30**	**35/45/57**	**53/64/72**	**64/73/79**	**71/79/84**	**88/91/93**	96/96/96	100/100/100	–/–/–	–/–/–
Fault 8	**1/1/2**	**1/1/3**	**2/2/4**	**10/10/15**	**19/43/59**	**56/61/72**	–/–/–	–/–/–	–/–/–	–/–/–
Fault 9	**1/1/23**	**1/5/47**	**7/30/64**	**23/45/73**	**30/52/77**	**37/59/81**	–/–/–	–/–/–	–/–/–	–/–/–
Fault 10	**7/7/10**	**1/1/3**	**1/1/4**	1/1/1	**1/1/3**	**10/10/13**	–/–/–	–/–/–	–/–/–	–/–/–
Fault 11	92/94/92	86/88/87	**46/47/51**	**20/21/26**	**32/32/34**	**57/57/58**	**93/94/94**	98/98/98	–/–/–	–/–/–
Fault 12	**1/1/4**	**1/1/4**	**1/1/3**	70/70/70	88/88/88	94/94/94	–/–/–	–/–/–	–/–/–	–/–/–
Fault 13	**1/1/5**	**1/1/2**	2/2/2	**8/8/10**	87/87/86	94/94/94	–/–/–	–/–/–	–/–/–	–/–/–
Fault 14	**1/1/11**	**3/3/30**	**14/15/47**	**31/38/58**	**49/50/60**	–/–/–	–/–/–	–/–/–	–/–/–	–/–/–

achieve a better reconstruction effect in the normal conditions more easily than AEs with the same structure. As for Sub-PCA models, the first two principal components are retained to construct the monitoring model.

After training the local models, the control limits of each phase are estimated to monitor the process. The confidence interval α is uniformly set to 99%. For the proposed method and AEs, control limits H^2 and SPE are estimated by KDE with Gaussian kernel. For Sub-PCA, control limits T^2 and SPE are estimated following the rules in (Nomikos and MacGregor, 1995).

The regenerated variable trajectories of cumulative base flow and cumulative acid flow are shown in Fig. 8. All three models can extract the main variation trend of the data set in each phase. However, the Sub-PCA model fails to follow nonlinear growth trends. For example, in phase 4, it is clear that Sub-PCA generates linear growth trajectories rather than first-order growth or recession trajectories while the NN based models capture the nonlinear growth trends. Moreover, the regenerated trajectories are more concentrated in the proposed LSTM-Encoder model than the AEs model, which indicates the LSTM structure can compensate the variations between different batches and generate a tighter control limit.

Remark 1. Note that the local models of different phases have the same network structure, so the expressivity of the local models is also unified. Hence, the varying training and validation losses among different phases are mainly influenced by the complexity of the dataset. The dataset complexity mainly includes the variation ranges in batch direction and the nonlinearity in the time direction. As can been seen from Fig. 8, phase 2 has a relatively bigger variation range in the batch direction and a rapid change in the time direction. Hence, the training and validation losses of phase 2 are bigger than other phases, which is consistent with the relatively greater dataset complexity.

4.2. *Process monitoring*

After process modeling, a dimension reduction model and the corresponding monitoring method with estimated control limits of each time interval in batches are constructed. The 50 normal validation batches and fault batches with different types, magnitudes and start times are tested to compare the process monitoring performance of Sub-PCA, AEs and the proposed LSTM-Encoder model. Fault detection rate (FDR) and false alarm rate(FAR) are selected to measure the performance following the definitions in Yin et al. (2012):

$$\begin{aligned} FDR &= \frac{No.\ of\ samples(J > J_{th} | f \neq 0)}{total\ samples(f \neq 0)} \\ FAR &= \frac{No.\ of\ samples(J > J_{th} | f = 0)}{total\ sample(f = 0)} \end{aligned} \tag{9}$$

where J and J_{th} are the control statistics and the corresponding threshold, $f = 0$ means the sample is normal, and $f \neq 0$ means the sample is a fault sample. Specifically, if either of the control statistics exceeds the corresponding thresholds, J is larger than J_{th}

Fig. 9 (a) is a typical monitoring result of a normal batch. It can be seen that the Sub-PCA, AEs, and the proposed LSTM-Encoder methods can all accurately monitor the normal samples. Besides, as can be seen in the H^2 control limits, the proposed method is more concentrated and smooth than the AEs, which is consistent with the hidden states and regeneration results listed in the last section.

Fig. 9(b) is an example of a fault batch monitoring result. Specifically, Fault 1 with −5% magnitude is chosen, and the fault starts after 352th sample points (the dash curve in the figure). It can be seen from the comparison that the proposed LSTM-Encoder detects the fault at the end of the third phase and the whole forth phase while the Sub-PCA and AEs model failed to detect the fault in a long duration of the fourth phase. Moreover, the exceeding control limit values are more significant in the proposed method.

Table 2 quantitatively summarizes the averaged FDRs for all fault batches with different fault magnitudes under a small averaged FARs (<2.5%). The fault numbers in Table 2 represent the magnitudes of faults, as listed in Table 1. It can be concluded from this table that the proposed LSTM-encoder methods can achieve higher FDRs in general. Especially in the faults with low and medium magnitudes, the other two methods can barely detect the faults while the proposed method gains a better performance. For the faults with high magnitudes that can be easily detected by the other two methods, the proposed method can gain the same detection result as the other two methods.

FDRs and FARs heavily depend on the choice of the confidence interval. The averaged FDRs and FARs of all faults under

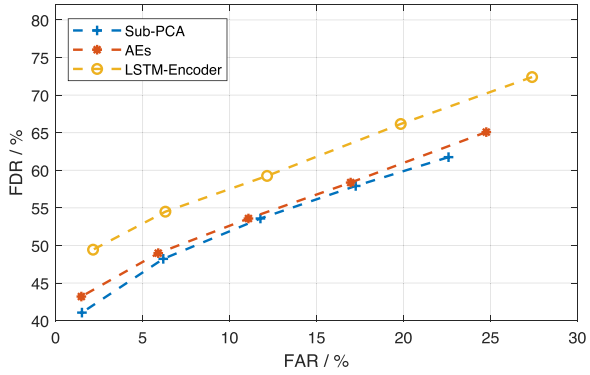

Fig. 10 – Averaged FDR and FAR results of all the fault batches under different confidence intervals.

different confidence intervals are shown in Fig. 10. It can be seen that the proposed method can gain a much higher FARs than AEs and Sub-PCA across the whole range, which indicates an intrinsic advantage over AEs and Sub-PCA in fault detection.

5. Conclusion

In this paper, a batch-wise LSTM-encoder decoder network that is suitable to extract the nonlinear dynamic features in the batch process and the corresponding process monitoring method is proposed. The LSTM-encoder is a multi-layer dimension reducing recurrent neural network. The input sequences are a series of samples from different batches. Based on the extracted hidden states and regenerated samples, the statistics H^2 and SPE are designed to measure the deviations in hidden state space and residual space. Then, control limits of the statistics H^2 and SPE are estimated for each time interval in a batch by kernel density estimation. Online process monitoring is achieved by comparing the statistics H^2 and SPE of the current sample with the corresponding control limits. A case study on an extensive reference penicillin fermentation dataset suggests that the proposed method can extract nonlinear dynamic features better and detect fault samples more effectively than previous methods.

Despite the superiority of the proposed method, it still has some weaknesses. The main weakness is the insufficient utilization of the within batch time-order characters. In the proposed method, the RNN cells only slide at the batch direction, and the encoder structure is used to capture the within batch characters. Hence, in the feature work, we want to focus on utilizing two-dimensional recurrent neural network methods like convolution LSTM to model the batch process more precisely.

Declaration of Competing Interest

The authors report no declarations of interest.

Acknowledgments

The authors would like to acknowledge the financial support from National Science Foundation China grant no. U1609213.

References

Adamowski, J., Chan, H.F., Prasher, S.O., Ozga-Zielinski, B., Sliusarieva, A., 2012. Comparison of multiple linear and nonlinear regression, autoregressive integrated moving average, artificial neural network, and wavelet artificial neural network methods for urban water demand forecasting in Montreal, Canada. Water Resour. Res. 48 (1).

Birol, G., Undey, C., Cinar, A., 2002. A modular simulation package for fed-batch fermentation: penicillin production. Comput. Chem. Eng. 26 (11), 1553–1565.

Bottou, L., 2010. Large-Scale Machine Learning With Stochastic Gradient Descent. Springer, pp. 177–186.

Breuel, T.M., 2015. Benchmarking of LSTM Networks. arXiv:1508.02774.

Glorot, X., Bordes, A., Bengio, Y., 2011. Deep sparse rectifier neural networks. Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, 315–323.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9 (8), 1735–1780.

Jia, M., Chu, F., Wang, F., Wang, W., 2010. On-line batch process monitoring using batch dynamic kernel principal component analysis. Chemometr. Intell. Lab. Syst. 101 (2), 110–122.

Jiang, Q.C., Yan, S.F., Yan, X.F., Yi, H., Gao, F.R., 2020. Data-driven two-dimensional deep correlated representation learning for nonlinear batch process monitoring. IEEE Trans. Ind. Informatics 16 (4), 2839–2848.

Kingma, D.P., Ba, J., 2014. Adam: A Method for Stochastic Optimization. arXiv:1412.6980.

Klambauer, G., Unterthiner, T., Mayr, A., Hochreiter, S., 2017. Self-normalizing neural networks. Advances in Neural Information Processing Systems, 971–980.

Lee, J.-M., Yoo, C., Lee, I.-B., 2004. Fault detection of batch processes using multiway kernel principal component analysis. Comput. Chem. Eng. 28 (9), 1837–1847.

Loy-Benitez, J., Heo, S., Yoo, C., 2020. Soft sensor validation for monitoring and resilient control of sequential subway indoor air quality through memory-gated recurrent neural networks-based autoencoders. Control Eng. Pract. 97, 104330.

Lu, N., Gao, F., Wang, F., 2004. Sub-PCA modeling and on-line monitoring strategy for batch processes. AIChE J. 50 (1), 255–259.

Mehta, B.R., Reddy, Y.J., 2015. OPC Communications. Butterworth-Heinemann, Oxford, pp. 459–477 (Chapter 15).

Nielsen, R.F., Nazemzadeh, N., Sillesen, L.W., Andersson, M.P., Gernaey, K.V., Mansouri, S.S., 2020. Hybrid machine learning assisted modelling framework for particle processes. Comput. Chem. Eng., 106916.

Nomikos, P., MacGregor, J.F., 1994. Monitoring batch processes using multiway principal component analysis. AIChE J. 40 (8), 1361–1375.

Nomikos, P., MacGregor, J.F., 1995. Multivariate SPC charts for monitoring batch processes. Technometrics 37 (1), 41–59.

Parzen, E., 1962. On estimation of a probability density function and mode. Ann. Math. Stat. 33 (3), 1065–1076.

Qin, S.J., 2012. Survey on data-driven industrial process monitoring and diagnosis. Annu. Rev. Control 36 (2), 220–234.

Shahnazari, H., 2020. Fault diagnosis of nonlinear systems using recurrent neural networks. Chem. Eng. Res. Des. 153, 233–245.

Van Impe, J., Gins, G., 2015. An extensive reference dataset for fault detection and identification in batch processes. Chemometr. Intell. Lab. Syst. 148, 20–31.

Wang, R.C., Baldea, M., Edgar, T.F., 2017. Data visualization and visualization-based fault detection for chemical processes. Processes 5 (3), 45.

Wang, R., Edgar, T.F., Baldea, M., Nixon, M., Wojsznis, W., Dunia, R., 2018. A geometric method for batch data visualization, process monitoring and fault detection. J. Process Control 67, 197–205.

Wang, K., Rippon, L., Chen, J., Song, Z., Gopaluni, R.B., 2019. Data-driven dynamic modeling and online monitoring for multiphase and multimode batch processes with uneven batch durations. Ind. Eng. Chem. Res.

Yan, W., Guo, P., gong, L., Li, Z., 2016. Nonlinear and robust statistical process monitoring based on variant autoencoders. Chemometr. Intell. Lab. Syst. 158, 31–40.

Yao, Y., Gao, F., 2009. A survey on multistage/multiphase statistical modeling methods for batch processes. Annu. Rev. Control 33 (2), 172–183.

Yin, S., Ding, S.X., Haghani, A., Hao, H., Zhang, P., 2012. A comparison study of basic data-driven fault diagnosis and process monitoring methods on the benchmark Tennessee Eastman process. J. Process Control 22 (9), 1567–1581.

Yu, W., Zhao, C., 2019. Robust monitoring and fault isolation of nonlinear industrial processes using denoising autoencoder and elastic net. IEEE Trans. Control Syst. Technol., 1–9.

Zhang, C., Song, D., Chen, Y., Feng, X., Lumezanu, C., Cheng, W., Ni, J., Zong, B., Chen, H., Chawla, N.V., 2019. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 1409–1416.

Zhao, C., Sun, Y., 2013. Step-wise sequential phase partition (sspp) algorithm based statistical modeling and online process monitoring. Chemometr. Intell. Lab. Syst. 125, 109–120.