Nonlinear dynamic soft sensor modeling with supervised long short-term memory network

Xiaofeng Yuan, *Member, IEEE*, Lin Li, Yalin Wang, *Member, IEEE*

***Abstract*—Soft sensor has been extensively utilized in industrial processes for prediction of key quality variables. To build an accurate virtual sensor model, it is very significant to model the dynamic and nonlinear behaviors of process sequential data properly. Recently, long short-term memory (LSTM) network has shown great modeling ability on various time series, in which basic LSTM units can handle data nonlinearities and dynamics with a dynamic latent variable structure. However, the hidden variables in the basic LSTM unit mainly focus on describing the dynamics of input variables, which lack representation for the quality data. In this paper, a supervised LSTM (SLSTM) network is proposed to learn quality-relevant hidden dynamics for soft sensor application, which is composed of basic SLSTM unit at each sampling instant. In the basic SLSTM unit, the quality and input variables are simultaneously utilized to learn the dynamic hidden states, which are more relevant and useful for quality prediction. The effectiveness of the proposed SLSTM network is demonstrated on a penicillin fermentation process and an industrial debutanizer column.**

***Index Terms*—Soft sensor; Quality prediction; Long short-term memory (LSTM); Supervised LSTM; Deep learning.**

I. Introduction

THE difficulty of timely measurement of many key quality variables has been a major problem for process monitoring, control and optimization in modern industry [1-4]. On-line hard devices or offline lab analyzers are often used to measure the quality variables in industrial processes. However, they often suffer from technical or economic limitations like severe environment, regular maintenance, expensive cost, large time delay, etc. Moreover, for some processes with short response times, they cannot meet the real-time requirements for process optimization and control.

To alleviate these problems, soft sensors have been constructed to give real-time estimation for the quality variables, which build inferential mathematical prediction models through those easy-to-measure process variables. Soft sensor was originated from the research of Brosilow et al. [5-7] for inference process control of multi-rate systems. The results showed that this inductive control method using estimator can greatly improve the control effect of process parameters and product quality. Since then, virtual sensors have been extensively utilized and successfully applied in many different process industries.

In the last decades, data-driven soft measurement technology has gained a lot of popularity since massive data becomes available with wide implementation of distributed control systems. Typical data-driven modeling methods include principal component analysis (PCA) [8], partial least squares (PLS) [9], etc. In order to deal with different process characteristics for soft sensors, researchers have further proposed different modeling strategies based on PCA, PLS and other models. For example, recursive PCA [10] and moving window PCA [11] are developed to deal with process time-varying problems. Also, support vector machine (SVM) [12] and artificial neural network (ANN) [13] are exploited to build soft sensors for nonlinear processes. Among them, ANN is one of the most popular methods that is widely applied in soft sensor. ANN consists of multiple layers of nonlinearities, which has potential advantages in dealing with large scale, high dimensional and highly nonlinear data patterns. However, ANN did not perform as well as expected in the past years because of the gradient vanishing or exploding problems when the network became very large. In the past decade, deep learning technique has been developed to overcome these problems and thus gained great success in many different areas. With the unsupervised layer-wise pretraining and fine-tune techniques, deep networks can be better trained. Moreover, hierarchical abstract features are learned from data for complex tasks like classification and regression.

Deep learning has also been introduced and exploited for inferential modeling of key quality variables in industrial processes [14-16]. As can be seen, most of these soft sensors are based on static deep networks like deep belief network (DBN) and stacked autoencoder (SAE). For these models, all the observed samples are assumed to be independent and identically distributed, in which the important temporal relationship is not considered. However, industrial processes are naturally dynamic with strong temporal correlations between data samples. In this way, process data are time series with high nonlinearities and dynamics. In order to model nonlinear dynamic processes, recurrent neural network (RNN) is widely used to describe temporal dynamic behavior for time sequences [17]. For example, Su et al. applied RNN to model and estimate the degree-of-cure in an epoxy/graphite fiber composites production process [18]. However, the standard RNN also suffers from the gradient vanishing and explosion problems, which makes it difficult to model long sequences. Thus, long short-term memory (LSTM) network has been developed to deal with this problem by Hochreiter et al. [19].

This work was supported in part by the Program of National Natural Science Foundation of China (61703440, 61590921, 61621062), in part by National Key R&D Program of China (2018YFB1701100), in part by the Natural Science Foundation of Hunan Province of China (2018JJ3687), and in part by Innovation-driven Plan in Central South University (2018CX011). (Corresponding author: Yalin Wang)

X. Yuan, L. Li and Y. Wang are with the School of Automation, Central South University, Changsha 410083,China (yuanxf@csu.edu.cn, 1192732314@qq.com, ylwang@csu.edu.cn).

As for the standard RNN, the commonly used sigmoid or tanh function is often used as the basic activation unit. Different from RNN, LSTM module is the basic activation unit at each temporal instant in LSTM network. With memory cells and three nonlinear gates in the basic LSTM unit structure, LSTM network is more effective in learning long-term temporal dependences since these memory cells can keep its state over long time and regulate the information flowing into and out of the cell. Recently, the LSTM network has been successfully used in many different fields such as emotion-sensitive artificial listening [20], automatic speech recognition [21], machine translation [22], etc. In industrial processes, LSTM has also been utilized for some soft sensor applications. For example, Sun et al. developed a structure with Gaussian-Bernoulli restricted Boltzmann machine and LSTM for quality prediction [23]. Although LSTM network can handle the dynamic problem of industrial processes, only input variables are utilized for dynamic feature learning in each LSTM unit. Each LSTM unit is unsupervised since they only focus on learning the dynamic hidden states from the input variables. However, the quality information is not considered and captured. There may be quality-irrelevant information in the hidden states. In this way, it cannot ensure the learned hidden states are relevant to the quality dynamics. In fact, the quality information is very important in supervised learning. Especially for soft sensor application, it would be more helpful to incorporate the quality information for supervised hidden feature learning from the input data. There already exist some supervised hidden variable models, like supervised probabilistic PCA, supervised linear dynamic system [24, 25]. In this paper, the quality information is exploited into the unsupervised LSTM cell to construct supervised LSTM (SLSTM) unit at each sampling instant. Then, SLSTM units is further connected to build deep SLSTM network for predictive model construction. In each SLSTM unit, both input and quality variables are utilized to learn the dynamic latent hidden states to largely keep their relevance with the quality information. In this way, the weight and bias terms of the SLSTM network are determined by both the input and quality variable information, which is beneficial for quality prediction.

The remaining sections of this paper is structured as follows. Section II provides a brief introduction about RNN and LSTM. Section III describes supervised LSTM in detail and the soft sensor modeling based on supervised LSTM network. Then, effectiveness and performance evaluation are validated on two case studies for the proposed method in Section IV. At last, conclusions and future work are given in Section V.

II. RNN AND LSTM

A. Recurrent Neural Network

Recurrent neural network is mainly used to describe the temporal dynamic behaviors of time sequential data. The network structure is given in Fig. 1 for RNN. The left illustrative structure can be unfolded as the right one, which consists of the input, hidden and output layers [26]. For the hidden layer states, each one is transmitted and connected to its next one.

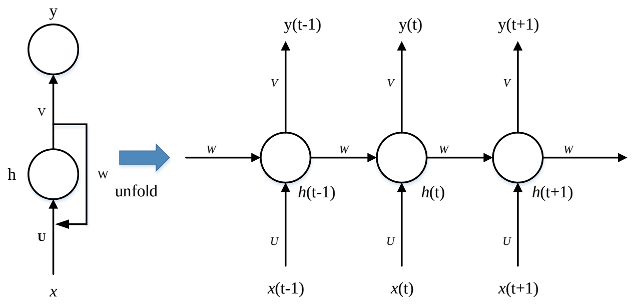


Fig. 1. The structure of the RNN

Let $x_{(t)}$, $h_{(t)}$ and $y_{(t)}$ represent the input, hidden and output vectors at sampling instant *t*, respectively. As can be seen from Fig. 1, the hidden and output vectors at sampling time *t* can be calculated as

$$h_{(t)} = \sigma(Ux_{(t)} + Wh_{(t-1)} + b) \quad (1)$$

$$y_{(t)} = \sigma(Vh_{(t)} + c) \quad (2)$$

where *b* and *c* represent the corresponding bias terms; $\sigma(\bullet)$ is the activation function like sigmoid function; *U*, *W* and *V* are the corresponding connection weights.The standard RNN has difficulties in learning long term dependencies, and is easily affected by the vanishing gradient problem [27].

B. Long Short-Term Memory

The long short-term memory (LSTM) network is a variant of the standard RNN. By replacing the basic hidden neurons with LSTM units in RNN, LSTM network can better handle the problem of gradient vanishing and explosion of long-term dependencies [28].

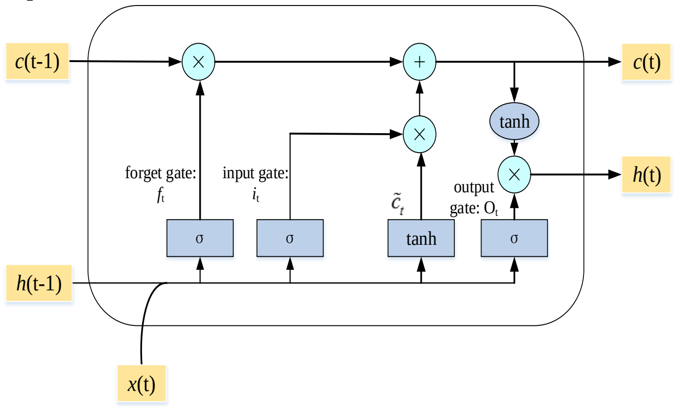


Fig. 2. The structure of the LSTM unit

The detailed structure is shown in Fig. 2 for the basic LSTM unit. Three gate controllers are placed into the LSTM unit, namely the input, forget and output gates. The three gates are mainly used to determine what information should be remembered. The LSTM network implements temporal memory through the switch of the gates to prevent the gradient vanishing. For the basic LSTM unit, its external inputs are its previous cell state $c_{(t-1)}$, the previous hidden state $h_{(t-1)}$ and the current input vector $x_{(t)}$.

From Fig. 2, the three gates are calculated as

$$f_{(t)} = \sigma(W_{fx}x_{(t)} + W_{fh}h_{(t-1)} + b_f) \quad (3)$$

$$i_{(t)} = \sigma(W_{ix}x_{(t)} + W_{ih}h_{(t-1)} + b_i) \quad (4)$$

$$o_{(t)} = \sigma(W_{ox}x_{(t)} + W_{oh}h_{(t-1)} + b_o) \tag{5}$$

where σ is the nonlinear activation function. Usually, the sigmoid function can be used as activation function for the gates. Inside the LSTM, an intermediate state $C_{(t)}$ is generated as

$$C_{(t)} = \tanh(W_{cx}x_{(t)} + W_{ch}h_{(t-1)} + b_c) \tag{6}$$

Then, the memory cell and hidden state of this LSTM are updated as

$$C_{(t)} = f_{(t)} \odot C_{(t)} + i_{(t)} \odot C_{(t)} \tag{7}$$

$$h_{(t)} = O_{(t)} \odot \tanh(C_{(t)}) \tag{8}$$

where tanh represents the nonlinear tanh activation function. $\odot$ is used to denote pointwise multiplication operation for two vectors.

III. Supervised LSTM

The LSTM network exploit a memory cell and gate structure to process and model long sequences, which can overcome the problem of gradient vanishing. However, the dynamic of quality variables is not learned for the hidden and cell states in the basic LSTM unit. As a matter of fact, it is very important to learn quality-related dynamic states for quality prediction in soft sensor applications. Hence, a supervised LSTM (SLSTM) network is proposed, which is composed of connected basic SLSTM units at each sampling instant. The structure of the basic SLSTM unit is shown in Fig. 3. As can be seen, the quality variable vector is additionally used as the input of the three gates and the intermediate cell. Thus, the quality information is exploited to guide the learning of hidden and cell states. Then, the SLSTM network can be trained by carrying out the forward pass and backpropagation through time (BPTT) iteratively.

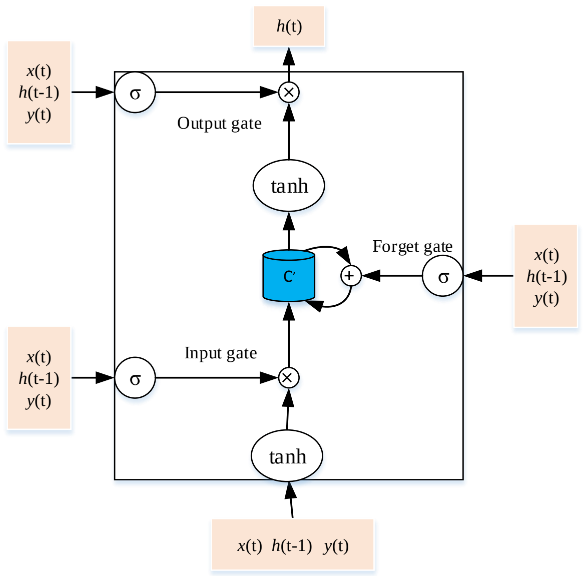


Fig. 3. The structure of the SLSTM unit

A. Forward Pass

Let N represent the number of SLSTM blocks, M and P be the dimensions of input vector $x_{(t)}$ and quality output vector $y_{(t)}$, respectively. Then, the following weights are used for a supervised LSTM layer.

1) Input weights:

$W_{ix}, W_{fx}, W_{ox}, W_{\tilde{c}x} \in \mathrm{R}^{N\times M}$

2) Recurrent weights:

$W_{ih}, W_{fh}, W_{oh}, W_{\tilde{c}h} \in \mathrm{R}^{N\times N}$

3) Output weights:

$W_{iy}, W_{fy}, W_{oy}, W_{\tilde{c}y} \in \mathrm{R}^{N\times P}$

4) Biases:

$b_i, b_o, b_c, b_f \in \mathrm{R}^{N}$

Then, the states of the three gate $f_{(t)}, i_{(t)}, o_{(t)}$, the intermediate cell $\tilde{c}_{(t)}$, the memory cell $c_{(t)}$ and the hidden state $h_{(t)}$ for a supervised LSTM unit can be written with forward pass as

$$neti = W_{ix}x_{(t)} + W_{ih}h_{(t-1)} + W_{iy}y_{(t)} + b_i \tag{9}$$

$$i_{(t)} = \sigma(neti) \tag{10}$$

$$netf = W_{fx}x_{(t)} + W_{fh}h_{(t-1)} + W_{fy}y_{(t)} + b_f \tag{11}$$

$$f_{(t)} = \sigma(netf) \tag{12}$$

$$neto = W_{ox}x_{(t)} + W_{oh}h_{(t-1)} + W_{oy}y_{(t)} + b_o \tag{13}$$

$$o_{(t)} = \sigma(neto) \tag{14}$$

$$net\tilde{c} = W_{\tilde{c}x}x_{(t)} + W_{\tilde{c}h}h_{(t-1)} + W_{\tilde{c}y}y_{(t)} + b_c \tag{15}$$

$$\tilde{c}_{(t)} = \sigma(net\tilde{c}) \tag{16}$$

$$c_{(t)} = f_{(t)} \odot c_{(t-1)} + i_{(t)} \odot \tilde{c}_{(t)} \tag{17}$$

$$h_{(t)} = o_{(t)} \odot \tanh(c_{(t)}) \tag{18}$$

After the forward pass is computed for the hidden state, it is used to predict the quality output with Eq. (2). The predicted quality output is calculated as

$$\hat{y}_{(t)} = \sigma(Vh_{(t)} + c) \tag{19}$$

where $\hat{y}_{(t)}$ is the predicted out of the t^{th} sample.

B. Backpropagation Through Time

Assume the training input and output data sequence are $\{x_{(1)}, x_{(2)}, ..., x_{(T)}\}$ and $\{y_{(1)}, y_{(2)}, ..., y_{(T)}\}$, where T is the total number of training data in the time sequence. After forward propagation, the following loss function is calculated for the training data as

$$E = \sum_{i=1}^{T}(y_{(t)} - \hat{y}_{(t)})^2 / T \tag{20}$$

Then, backpropagation through time can be used to tune the network parameters.

The gradient deltas can be computed inside the SLSTM unit as

$$\begin{aligned}\delta h_{(t)} = \Delta^t + \delta o_{(t+1)}W_{oh}^T + \delta i_{(t+1)}W_{ih}^T + \delta f_{(t+1)}W_{fh}^T \\ + \delta\tilde{c}_{(t+1)}W_{\tilde{c}h}^T\end{aligned} \tag{21}$$

$$\delta o_{(t)} = (\delta h_{(t)})^T \odot \tanh(c_{(t)}) \odot \sigma'(neto) \tag{22}$$

$$\begin{aligned}\delta\tilde{c}_{(t)} = (\delta h_{(t)})^T \odot o_{(t)} \odot (1 - \tanh(c_{(t)})^2) \odot i_{(t)} \\ \odot \tanh'(net\tilde{c})\end{aligned} \tag{23}$$

$$\delta f_{(t)} = (\delta h_{(t)})^T \odot o_{(t)} \odot (1-\tanh(c_{(t)})^2) \odot c_{(t-1)} \odot \sigma'(netf) \tag{24}$$

$$\delta i_{(t)} = (\delta h_{(t)})^T \odot o_{(t)} \odot (1-\tanh(c_{(t)})^2) \odot \tilde{c}_{(t)} \odot \sigma'(neti) \tag{25}$$

where Δ^t represents the deltas vector from the previous high layer, which corresponds to ($\partial E / \partial h^t$). Here, E is just the loss function in Eq. 20. $\sigma'(\bullet)$ and $\tanh'(\bullet)$ represent the derivatives of the corresponding function. $(\bullet)^T$ represents the transposition of a vector or matrix.

Finally, the following gradients can be calculated for the weights as

$$\delta W_{*x} = \sum_{t=1}^{T} \left\langle \delta *_{(t)}, x_{(t)} \right\rangle \tag{26}$$

$$\delta W_{*h} = \sum_{t=0}^{T-1} \left\langle \delta *_{(t+1)}, h_{(t)} \right\rangle \tag{27}$$

$$\delta W_{*y} = \sum_{t=1}^{T} \left\langle \delta *_{(t)}, y_{(t)} \right\rangle \tag{28}$$

$$\delta b_{*} = \sum_{t=1}^{T} \delta *_{(t)} \tag{29}$$

where $*$ is a certain term of f, i, o, c; also, $< *_1, *_2 >$ is used to denote the exterior product between two vectors. For training of SLSTM model, the Adam algorithm is used since it has advantages over the momentum gradient descent method and the root mean square back propagation (RMSProp) algorithm, which can calculate the adaptability of different parameters while consume less processor resources [29]. The Adam algorithm can be written as

$$v_w = \beta_1 v_w + (1-\beta_1)\delta W \tag{30}$$

$$v_b = \beta_1 v_b + (1-\beta_1)\delta b \tag{31}$$

$$s_w = \beta_2 s_w + (1-\beta_2)\delta W \odot \delta W \tag{32}$$

$$s_b = \beta_2 s_b + (1-\beta_2)\delta b \odot \delta b \tag{33}$$

$$v_w^c = v_w \big/ \left(1-\beta_1\right) \tag{34}$$

$$s_w^c = s_w \big/ \left(1-\beta_2\right) \tag{35}$$

$$v_b^c = v_b \big/ \left(1-\beta_1\right) \tag{36}$$

$$s_b^c = s_b \big/ \left(1-\beta_2\right) \tag{37}$$

$$W := W - \alpha v_w \Big/ \left(\sqrt{s_w^c} + \varepsilon\right) \tag{38}$$

$$b := b - \alpha v_b \Big/ \left(\sqrt{s_b^c} + \varepsilon\right) \tag{39}$$

where v_w and v_b are the first moment estimation of the gradient and bias; δW and δb are the gradients for the weights and biases; s_w and s_b are the second moment estimation of the gradient and bias; β_1 and β_2 are exponential decay rates, with default values as 0.9 and 0.999, respectively; v_*^c and s_*^c indicate deviation correction for the gradient mean of the weights and biases, respectively, where $*$ can be W or b ; ε is a small constant for numerical stability, the default value is 10^{-8} ; Step size α defaults to 0.001.

In the supervised LSTM network, $h_{(t-1)}$, $x_{(t)}$ and $y_{(t)}$ are combined as inputs for the basic LSTM unit to learn hidden states that are used for quality prediction. Then, the weight and bias parameters of the network are obtained by the BPTT algorithm. Hence, the quality variables are used to guide the learning of hidden and cell states, which would also have information on the weight and bias parameters of the network.

It can be known from Eq. 9-16 that the unit $c_{(t)}$ implements the memory of long-term sequence information. In $h_{(t)}$, the predicted output information of the current moment is stored. In this way, the network retains the capability of processing long term sequences, and ensure the accuracy of prediction, simultaneously.

The basic SLSTM unit can be used to construct the deep SLSTM network, the structure of which is provided in Fig. 4. With the deep SLSTM network, deep nonlinear dynamic quality-relevant hidden features can be learned from low to high layers.

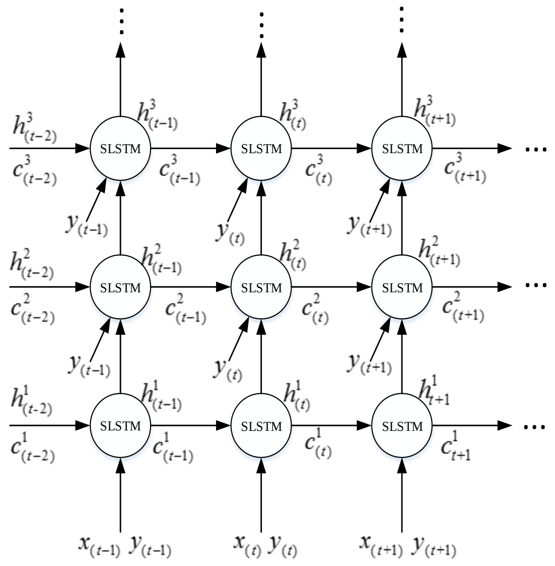


Fig. 4. The structure of the deep SLSTM network

C. Supervised LSTM Based Soft Sensor

SLSTM can be easily applied to build soft sensor network. Fig. 5 gives the diagram for the SLSTM-based soft sensor modeling framework. The main steps are as follows.

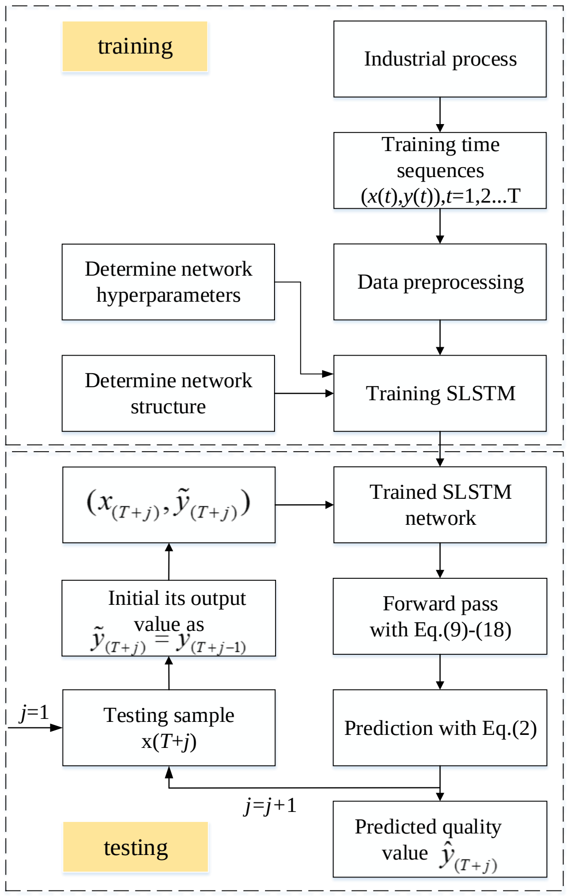

training
Industrial process
Training time sequences (x(t),y(t)),t=1,2...T
Determine network hyperparameters
Data preprocessing
Determine network structure
Training SLSTM
$(x_{(T+j)}, \tilde{y}_{(T+j)})$
Trained SLSTM network
Initial its output value as $\tilde{y}_{(T+j)} = y_{(T+j-1)}$
Forward pass with Eq.(9)-(18)
j=1
Testing sample x(T+j)
Prediction with Eq.(2)
j=j+1
Predicted quality value $\hat{y}_{(T+j)}$
testing

Fig. 5. The flowchart of the proposed SLSTM-based soft sensor model

1) Collect training time series from a specific process, for which the input and output sequences are $\{x_{(1)}, x_{(2)}, ..., x_{(T)}\}$ and $\{y_{(1)}, y_{(2)}, ..., y_{(T)}\}$. Normalize data before carrying out modeling.

2) Determine the structure and hyperparameters of the SLSTM network, and then train it on $\{x_{(1)}, x_{(2)}, ..., x_{(T)}\}$ and $\{y_{(1)}, y_{(2)}, ..., y_{(T)}\}$.

3) After training stage, it can be used to predict the output for the subsequent testing samples one by one.

4) Assume the input of the testing time sequence is $\{x_{(T+j)}\}_{j=1,2,...,T_{testing}}$. To estimate the quality value of $x_{(T+j)}$ with the SLSTM network, the output part is unknown for the network. Thus, an initialization strategy is utilized to firstly give a rough output estimation. For each $x_{(T+j)}$ from the first to the last sample in the testing sequence, its output value is initialized with that of its previous sample. That is to say, the initial value is set as $\tilde{y}_{(T+j)} = y_{(T+j-1)}$.

5) The initialized data sample is then substituted into the trained SLSTM network. By forward pass with Eq. 9-18, the hidden states can be obtained for $x_{(T+j)}$.

6) At last, the final output is estimated as $\hat{y}_{(T+j)}$ for $x_{(T+j)}$ with Eq. 2.

To assess model effectiveness, the root mean squared error (RMSE) is often exploited for performance evaluation, which are defined for the training and testing datasets as

$$RMSE_{training} = \sqrt{\sum_{t=1}^{T} (y_{(t)} - \hat{y}_{(t)})^2 / T} \tag{40}$$

$$RMSE_{testing} = \sqrt{\sum_{t=T+1}^{T+T_{testing}} (y_{(t)} - \hat{y}_{(t)})^2 / T_{testing}} \tag{41}$$

where $y_{(t)}$ and $\hat{y}_{(t)}$ are the labeled and estimated quality values at sampling time t, respectively. T and $T_{testing}$ are the total number of data samples in the training and testing sets.

IV. Case Studies

To validate the performance of SLSTM, it is applied for virtual sensor on a debutanizer column and penicillin fermentation process.

A. *Debutanizer Column*

Debutanizer is an important part of desulfurization and naphtha separation units in petroleum production processes. Fig. 6 shows the basic flowchart of the debutanizer column. It is mainly used to separate C5 from C3 and C4. It removes C3 and most C4 from the top, as well as C5 and a small amount of residual C4 at the bottom. Since the bottom butane concentration has a great influence on the debutanizer column and it is difficult to measure it directly, it needs to be strictly detected and controlled. For the debutanizer column, it is important and necessary to provide real-time prediction of the butane concentration for effective control and optimization. Thus, virtual sensors are exploited to predict the butane concentration. To predict the butane concentration of the process, seven routinely measured process variables that are highly related with the quality variable has been selected as the inputs for virtual sensors. Table I gives a detailed description for the seven variables. In total, 2394 samples were collected in this process. The dataset is partitioned into two parts: about two thirds for training dataset (1556 samples) and the remaining one third for testing dataset (837 samples).

As can be seen, $x_{(t)}$ and $y_{(t)}$ in the SLSTM unit are with dimensions of 7 and 1, respectively. The network structure with the number of hidden layers and neurons is determined with trial and error technique by carrying out different layers and neurons. In this process, the SLSTM network is set with one hidden layer of 90 neurons.

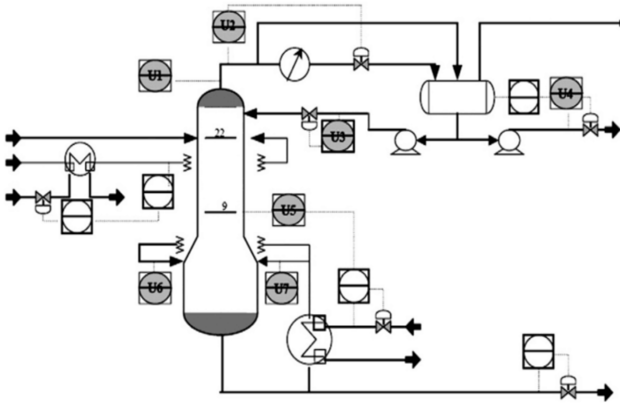


Fig. 6. Flowchart for the debutanizer column[30]

TABLE I Secondary Variables for soft sensor on the debutanizer column

Secondary variables	Description
u_1	Top temperature
u_2	Top pressure
u_3	Flow of reflux
u_4	Flow to the next process
u_5	Temperature of the sixth tray
u_6	Temperature A at bottom
u_7	Temperature B at bottom

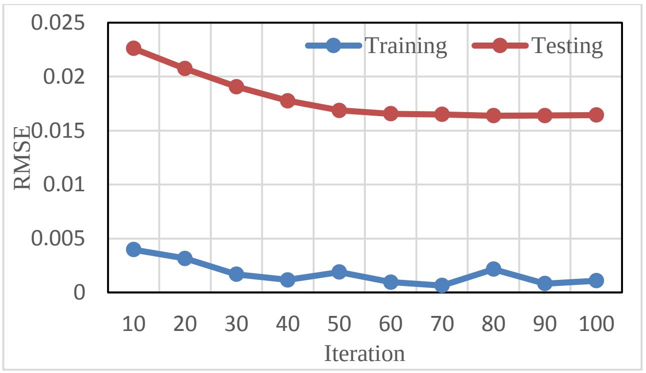


Fig. 7 Comparison of RMSE with different iterations for SLSTM on the debutanizer column

To train the SLSTM network, it is important to determine another key hyperparameter for the number of iterations in BPTT. Here, the prediction performance of RMSE with different iterations is investigated for both the training and testing datasets. The number of training iterations is selected from range set [10 20 30 40 50 60 70 80 90 100]. With different iterations, the RMSE is calculated for both the training and testing datasets. The results are given in Fig. 7 for the SLSTM-based soft sensor network. As can be seen, the general RMSE trend gets decreased with the increase of the iteration number both for the training and testing datasets. Moreover, the RMSE reaches the convergent state when the iteration is about 60. Thus, the number of the iterations is selected as 60 in this study.

TABLE II Training and testing RMSE of the three methods on the debutanizer column

Method	RNN	LSTM	SLSTM
Training RMSE	0.13262	0.12793	0.00218
Testing RMSE	0.18071	0.17829	0.01599

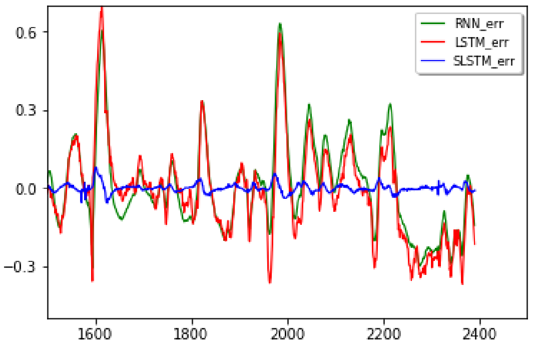
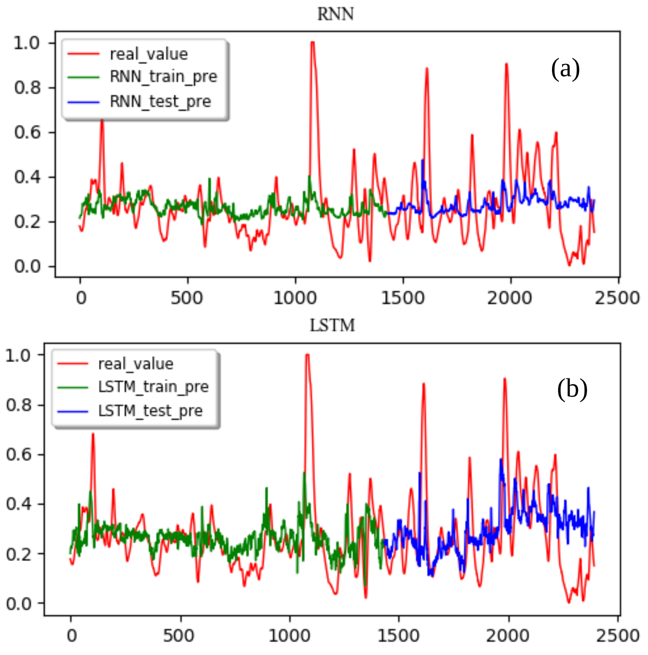


Fig. 8. Comparison of prediction errors of RNN, LSTM and SLSTM on the debutanizer column

For performance analysis, the standard RNN, LSTM and SLSTM models are developed and compared for quality prediction. The RMSE values with the three soft sensors are compared in Table II on both datasets. From Table II, RNN gives the worst prediction performance on both datasets. For the standard RNN, it uses the basic tanh unit to learn the dynamic hidden states, which only considers the states at the recent moments. Hence, it is very difficult to learn the long-time dependencies for output prediction. Differently, LSTM utilizes a memory cell to store useful long-term status information. In this way, LSTM can get better performance than RNN. However, the LSTM basic cell mainly learns dynamic hidden states from the input variables, which does not include important dynamic quality information. By introducing the quality variable to the inputs of the basic SLSTM unit, it can learn useful dynamic information that are relevant to the quality variables. Thus, it is more suitable for quality prediction. The RMSE values of SLSTM are 0.00218 and 0.01599 on the training and testing data, respectively, which are much smaller than these of RNN and LSTM networks. Hence, SLSTM has much higher prediction accuracy since the quality variable is introduced for quality-relevant dynamic feature learning. Fig. 8 shows the detailed prediction errors of RNN, LSTM and SLSTM on the testing dataset, respectively. It is easily seen the predicted errors of RNN and LSTM network are mostly between range [-0.4 0.6]. However, SLSTM have much smaller prediction errors that are mostly around zero.

Furthermore, the detailed sample predictions on the training and testing datasets are further visualized in Fig. 9 for RNN, LSTM and SLSTM. From Fig. 9, the prediction curves of SLSTM can track very well with the real output curves both in the training and testing datasets. Differently, large deviations exist between the real and predicted output curves for RNN and LSTM. Therefore, the prediction accuracy is largely improved by introducing the quality variable into LSTM for supervised learning. It should be noted that SLSTM has almost exact predictions with the real quality values. Hence, the predicted and labeled lines overlap so much that it seems there is only one line.



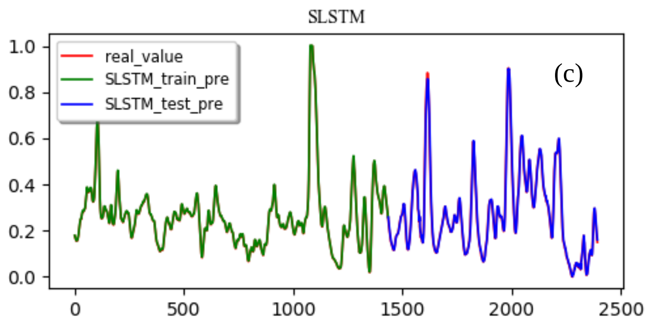


Fig. 9. Detailed Predictions of RNN, LSTM and SLSTM on the debutanizer column: (a) RNN; (b) LSTM; (c) SLSTM

B. Penicillin Fermentation Process

The penicillin fermentation is a very complex biochemical batch process to make product useful to humans, which is accompanied with complicated secondary microbial metabolism. The fermentation rate often depends on the concentration of cells, microorganisms, cellular components, temperature, pH, oxygen content, etc. Product recovery frequently involves the concentration of the dilute solution. A Pensim Simulator is provided for this process as a benchmark for evaluation of different modeling, process monitoring and control strategies. The simulator can be accessed at website: http://simulator.iit.edu/web/pensim/index.html. The basic flow sheet is shown in Fig. 10 for the penicillin fermentation process[31].

For effective process optimization and control, it is of great significance to establish a predictive model of its key variables like the concentration of the penicillin. Thirteen variables are selected as secondary variables for the soft sensor model. Table III provides a detailed description of these variables. To model the process nonlinear dynamic behaviors, SLSTM-based soft sensing strategy is adopted in this study.

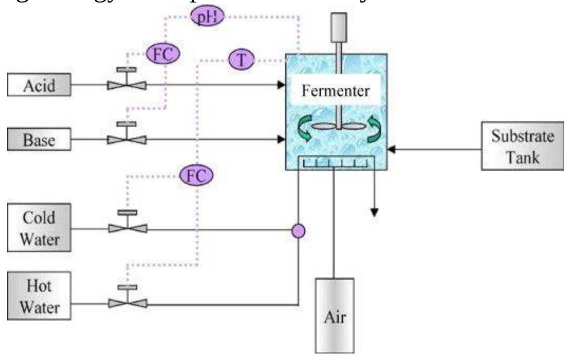


Fig. 10. Flowchart of the penicillin fermentation process [32]

TABLE III Input description of the fermentation process

Input variables	Variable description
u_1	Agitator power
u_2	Aeration rate
u_3	Substrate feed temperature
u_4	Substrate feed rate
u_5	Cooling water flow rate
u_6	Base flow rate
u_7	Dissolved oxygen concentration
u_8	Substrate concentration
u_9	Culture volume
u_{10}	Biomass concentration
u_{11}	PH
u_{12}	Carbon dioxide concentration
u_{13}	Fermenter temperature

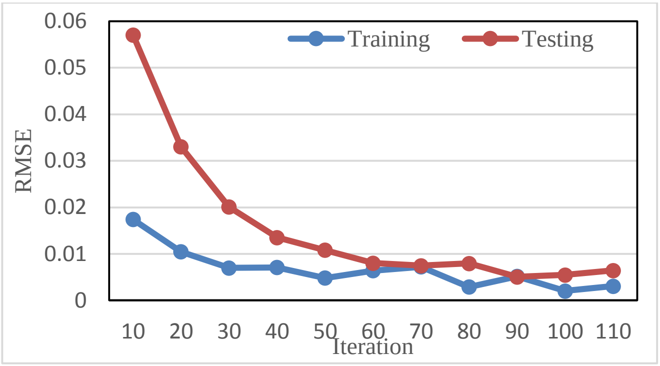
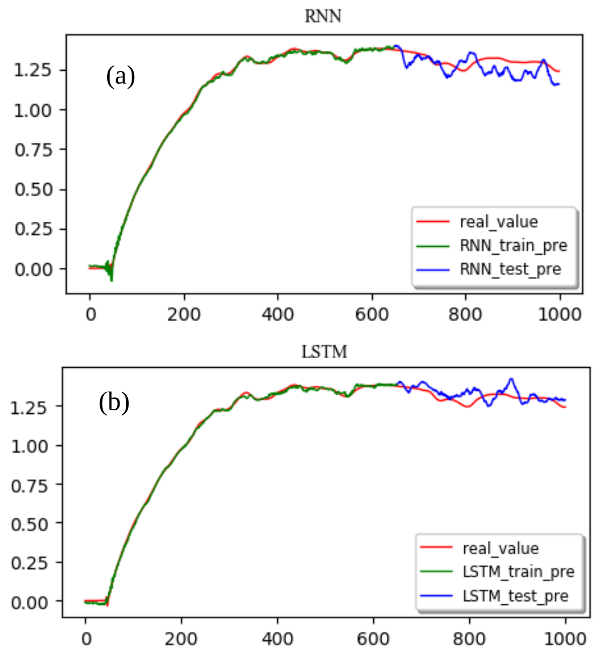


Fig. 11 Comparison of RMSE with different iterations for SLSTM on the penicillin process

1000 samples were simulated in the penicillin process. Among them, the first 650 samples are taken as training dataset, and the remaining ones are for testing purpose. Input vector $x_{(t)}$ and output vector $y_{(t)}$ in the SLSTM unit are with dimensions of 13 and 1, respectively. By trial and error, the network is configured with a hidden structure of [90 70].

To train the SLSTM based soft sensor model, the number of iterations is investigated for the BPTT algorithm again. Here, the prediction performance of RMSE with different iterations is calculated for both the training and testing datasets. The number of training iterations is selected from set [10 20 30 40 50 60 70 80 90 100 110]. Fig. 11 provides the detailed results. From Fig. 11, when the number of iterations is small, the prediction RMSE decreases rapidly with the increase of the iterations. With iteration value at 100, the RMSE reaches at relatively small states both for the training and testing datasets. Then, it no longer gets greatly improved when the iteration becomes large. Therefore, the number of the iterations is determined as 100 in this case.



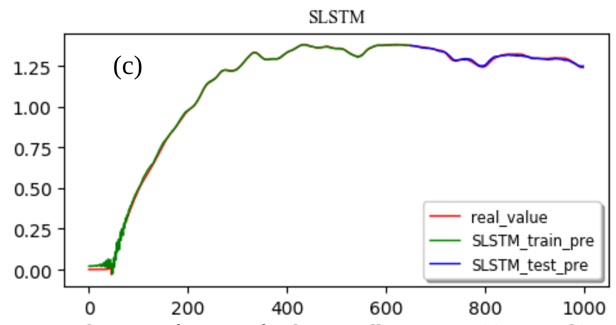


Fig. 12. Prediction performance for the penicillin process: (a) RNN; (b) LSTM; (c) SLSTM.

TABLE IV Prediction RMSE of LSTM and SLSTM

Method	RNN	LSTM	SLSTM
Training RMSE	0.01213	0.01127	0.01101
Testing RMSE	0.07288	0.04507	0.00627

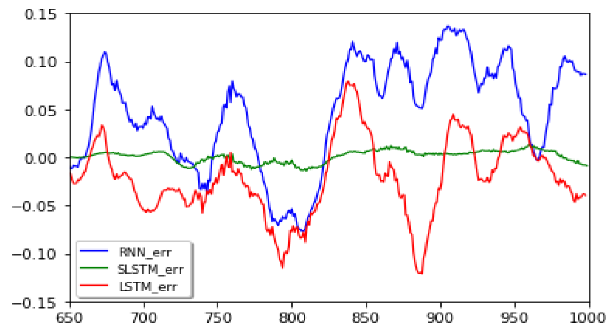


Fig. 13. Comparison of prediction error of RNN, LSTM and SLSTM

For performance comparison, RNN, LSTM and SLSTM models are developed to predict the penicillin concentration. Table IV gives the prediction results of the three methods on both the training and testing datasets. Similar to the debutanizer column, RNN shows the worst prediction accuracy on both datasets since RNN is difficult to remember long-term information. While for LSTM, it can learn the long-term input dependencies by adopting the memory cell. Therefore, LSTM outperforms the standard RNN a little. However, both the basic units in RNN and LSTM are essentially unsupervised learning modules, in which the dynamic states are mainly learned from the input variables. Hence, there may be quality-irrelevant information in these dynamic states. Differently, the quality variables are introduced to the basic SLSTM unit, it can largely learn important quality-relevant dynamic information for output prediction. From the prediction RMSE, SLSTM can largely improve the prediction accuracy with smaller RMSE values both on the training and testing datasets. Fig. 12 shows the detailed prediction values for the training and testing datasets with the RNN, LSTM and SLSTM. As can be seen, the proposed SLSTM method is superior to the traditional RNN and LSTM in tracking with the real trend of the output variable. The prediction of the proposed SLSTM method is in good match with the actual trajectory of penicillin concentration, and has a much smaller deviation than RNN and LSTM. Fig. 13 further shows the detailed prediction errors for RNN, LSTM and SLSTM on the testing dataset, respectively. The prediction error of proposed SLSTM is very small among the entire testing dataset. Hence, SLSTM is more suitable to describe the dynamic behavior of the quality variable, which is able to improve the prediction performance largely.

V. Conclusion

In this paper, a SLSTM network is proposed for nonlinear dynamic modeling for soft sensor applications. In the basic SLSTM module, both the input and quality variables are utilized to learn the shared dynamic hidden feature states. In this way, the hidden states contain quality-relevant dynamic information for the quality variables, which is more beneficial for quality regression prediction. The SLSTM units can be easily utilized to construct deep networks for hierarchical nonlinear dynamic hidden feature description. Applications on the debutanizer column and penicillin fermentation process show the effectiveness of the SLSTM-based soft sensor model. Experiment results demonstrate that the prediction performance of SLSTM outperforms RNN and LSTM. For the SLSTM, it needs a little more computational resource and training time since the quality variable is added to each LSTM unit. For further research, it can be taken into consideration of the correlation influence of different input variables on the quality variable.

References

[1] M. Kano, and Y. Nakagawa, "Data-based process monitoring, process control, and quality improvement: Recent developments and applications in steel industry," *Comput. Chem. Eng.*, vol. 32, no. 1, pp. 12-24, 2008.
[2] P. Kadlec, R. Grbić, and B. Gabrys, "Review of adaptation mechanisms for data-driven soft sensors," *Comput. Chem. Eng.*, vol. 35, no. 1, pp. 1-24, 2011.
[3] S. Khatibisepehr, B. Huang, and S. Khare, "Design of inferential sensors in the process industry: A review of Bayesian methods," *J. Process Contr.*, vol. 23, no. 10, pp. 1575-1596, 11, 2013.
[4] Z. Ge, "Process data analytics via probabilistic latent variable models: A tutorial review," *Ind. Eng. Chem. Res.*, vol. 57, pp. 12646-12661, 2018.
[5] B. Joseph, and C. Brosilow, "Inferential Control of Processes: 3. Construction of Optimal and Suboptimal Dynamic Estimators," *AIChE J.*, vol. 24, no. 3, pp. 500-509, 2010.
[6] B. Joseph, and C. B. Brosilow, "Inferential Control of Processes: I. Steady State Analysis and Design," *AIChE J.*, vol. 24, no. 3, pp. 485-492, 2010.
[7] M. Tabor, J. J. Chae, G. D. Burnett, and D. J. Durian, "Inferential control of processes: Part II. The structure and dynamics of inferential control systems," *AIChE J.*, vol. 24, no. 3, pp. 492-500, 2010.
[8] X. Yuan, Z. Ge, B. Huang, Z. Song, and Y. Wang, "Semisupervised JITL Framework for Nonlinear Industrial Soft Sensing Based on Locally Semisupervised Weighted PCR," *IEEE. T. Ind. Inf.*, vol. 13, no. 2, pp. 532-541, 2017.
[9] M. Kano, K. Miyazaki, S. Hasebe, and I. Hashimoto, "Inferential control system of distillation compositions using dynamic partial least squares regression," *J. Process Contr.*, vol. 10, no. 2, pp. 157-166, 2000.
[10] W. Li, H. H. Yue, S. Valle-Cervantes, and S. J. Qin, "Recursive PCA for adaptive process monitoring," *J. Process Contr.*, vol. 10, no. 5, pp. 471-486, 2000.
[11] M. Ammiche, A. Kouadri, and A. Bensmail, "A Modified Moving Window dynamic PCA with Fuzzy Logic Filter and application to fault detection," *Chemometr. Intell. Lab. Syst*, vol. 177, pp. 100-113, 2018.
[12] H. Kaneko, and K. Funatsu, "Application of online support vector regression for soft sensors," *AIChE J.*, vol. 60, no. 2, pp. 600-612, 2014.
[13] L. Fortuna, P. Giannone, S. Graziani, and M. G. Xibilia, "Virtual instruments based on stacked neural networks to improve product quality monitoring in a refinery," *IEEE T. Instrum. Meas.*, vol. 56, no. 1, pp. 95-101, 2007.
[14] C. Shang, F. Yang, D. Huang, and W. Lyu, "Data-driven soft sensor development based on deep learning technique," *J. Process Contr.*, vol. 24, no. 3, pp. 223-233, 2014.
[15] X. Yuan, B. Huang, Y. Wang, C. Yang, and W. Gui, "Deep learning based feature representation and its application for soft sensor modeling

with variable-wise weighted SAE," *IEEE. T. Ind. Inf.*, vol. 14, no. 7, pp. 3235-3243, 2018.

[16] B. Andò, S. Graziani, and M. G. Xibilia, "Low-order Nonlinear Finite-Impulse Response Soft Sensors for Ionic Electroactive Actuators Based on Deep Learning," *IEEE T. Instrum. Meas.*, pp. 10.1109/TIM.2018.2884450, 2019.

[17] S. R. Vijaya Raghavan, T. K. Radhakrishnan, and K. Srinivasan, "Soft sensor based composition estimation and controller design for an ideal reactive distillation column," *ISA T.*, vol. 50, no. 1, pp. 61-70, 2011.

[18] H. B. Su, L. T. Fan, and J. R. Schlup, "Monitoring the process of curing of epoxy/graphite fiber composites with a recurrent neural network as a soft sensor," *Eng. Appl. Artif. Intel*, vol. 11, no. 2, pp. 293-306, 1998.

[19] S. Hochreiter, and J. Schmidhuber, "Long Short-Term Memory," *Neural. Comput.*, vol. 9, no. 8, pp. 1735-1780, 1997.

[20] M. Wollmer, B. Schuller, F. Eyben, and G. Rigoll, "Combining Long Short-Term Memory and Dynamic Bayesian Networks for Incremental Emotion-Sensitive Artificial Listening," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, no. 5, pp. 867-881, 2010.

[21] M. Wӧllmer, F. Weninger, J. Geiger, B. Schuller, and G. Rigoll, "Noise robust ASR in reverberated multisource environments applying convolutive NMF and Long Short-Term Memory ☆," *Computer Speech & Language*, vol. 27, no. 3, pp. 780-797, 2013.

[22] N. Srivastava, E. Mansimov, and R. Salakhutdinov, "Unsupervised Learning of Video Representations using LSTMs," 2015.

[23] Q. Sun, and Z. Ge, "Probabilistic Sequential Network for Deep Learning of Complex Process Data and Soft Sensor Application," *IEEE. T. Ind. Inf.*, pp. DOI: 10.1109/TII.2018.2869899 2018.

[24] Z. Ge, and X. Chen, "Dynamic Probabilistic Latent Variable Model for Process Data Modeling and Regression Application," *IEEE. T. Contr. Syst. T.*, vol. 29, pp. 323 - 331, 2019.

[25] X. Yuan, Z. Ge, Z. Song, Y. Wang, C. Yang, and H. Zhang, "Soft Sensor Modeling of Nonlinear Industrial Processes Based on Weighted Probabilistic Projection Regression," *IEEE T. Instrum. Meas.*, vol. 66, no. 4, pp. 837-845, 2017.

[26] R. J. Williams, and D. Zipser, "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks," *Neural. Comput.*, vol. 1, no. 2, pp. 270-280, 2014.

[27] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training Recurrent Neural Networks," in International Conference on Machine Learning, 2013, pp. 1310-1318.

[28] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A Search Space Odyssey," *IEEE Transactions on Neural Networks & Learning Systems*, vol. 28, no. 10, pp. 2222-2232, 2015.

[29] D. Kingma, and J. Ba, "Adam: A Method for Stochastic Optimization," *Comput. Sci.*, 2014.

[30] Fortuna, and Luigi, "Soft Sensors for Monitoring and Control of Industrial Processes," *Advances in Industrial Control*, 2006.

[31] G. Birol, C. Ündey, and A. Cinar, "A modular simulation package for fed-batch fermentation: penicillin production," *Comput. Chem. Eng.*, vol. 26, no. 11, pp. 1553-1565, 2002.

[32] X. Yuan, Z. Ge, and Z. Song, "Locally Weighted Kernel Principal Component Regression Model for Soft Sensing of Nonlinear Time-Variant Processes," *Ind. Eng. Chem. Res.*, vol. 53, no. 35, pp. 13736-13749, 2014.

Xiaofeng Yuan received the B.Eng. and Ph.D. degrees from the Department of Control Science and Engineering, Zhejiang University, Hangzhou, China, in 2011 and 2016, respectively.

He was a visiting scholar with the Department of Chemical and Materials Engineering, University of Alberta, Edmonton, AB, Canada, from November 2014 to May 2015. He is currently an Associate Professor with the School of Automation, Central south University. His research interests include deep learning and artificial intelligence, machine learning and pattern recognition, industrial process soft sensor modeling, process data analysis, etc.

Lin Li is a postgraduate student at the School of Automation, Central South University, Changsha, China. She received her B.Eng. degree in School of Information Engineering from Xiangtan University, Xiangtan, China, in 2018. Her research interests include deep learning, machine learning, soft sensor modeling, process data mining, etc.

Yalin Wang received the B.Eng. and Ph.D. degrees from the Department of Control Science and Engineering, Central South University, Changsha, China, in 1995 and 2001, respectively.

Since 2003, she has been with the School of Information Science and Engineering, Central south University, where she was at first an Associate Professor and then a Professor. She is currently a Professor with the School of Automation, Central South University. Her research interests include the modeling, optimization and control for complex industrial processes, intelligent control, and process simulation.