Supervised deep belief network for quality prediction in industrial processes

Article · November 2020

DOI: 10.1109/TIM.2020.3035464

3 authors:

Xiaofeng Yuan
Central South University
68 PUBLICATIONS **824** CITATIONS
SEE PROFILE

Yongjie Gu
Central South University
2 PUBLICATIONS **2** CITATIONS
SEE PROFILE

Yalin Wang
Central South University
105 PUBLICATIONS **763** CITATIONS
SEE PROFILE

Supervised deep belief network for quality prediction in industrial processes

Xiaofeng Yuan, *Member*, *IEEE*, Yongjie Gu, Yalin Wang, *Member*, *IEEE*

***Abstract*—Deep belief network (DBN) has been recently applied for soft sensor modeling with its excellent feature representation capacity. However, DBN cannot guarantee the extracted features are quality-related and beneficial to further quality prediction. To solve this problem, a novel supervised deep belief network (SDBN) is proposed in this paper by introducing the quality information into the training phase. SDBN consists of multiple supervised restricted Boltzmann machines (SRBM) with a stacked structure. In each SRBM, the quality variables are added to the visible layer for network pre-training and feature learning. Thus, the pretrained weights can act as better initializations for the whole network for fine-tuning. Moreover, it can ensure that the learned features are largely quality-related for soft sensor. Finally, the SDBN-based soft sensor model is applied to two industrial plants of a debutanizer column and a hydrocracking process for quality prediction.**

***Index Terms*—DBN; Soft sensor; Quality prediction; SDBN; SRBM.**

I. INTRODUCTION

The rapid development of science and technology has put forward more stringent requirements on modern industrial processes like petrochemical, steelmaking, papermaking and metallurgical industries. In order to achieve the goal of refinement, standardization, intellectualization of modern production, advanced process control (APC) are adopted to optimize the process for increased product yields and reduced utility costs. Basically, APC strategies are mostly based on accurate measurement and monitoring of process running data, especially of the key quality variables like chemical components and quality attributes. However, due to reasons like the harsh on-site environment, expensive instrument cost and large measurement delay, most key quality variables are difficult to measure online to provide real-time feedback information for process control strategies. Differently, there are many easy-to-measure routine process variables like temperatures, pressures, flow-rates and liquid levels, which can be easily measured by online physical sensors like thermocouples. Thus, soft sensor technique has been developed to predict the quality variables on the basic idea of inferential control [1-4]. At present, the technology is widely used in petrochemical, steelmaking, papermaking, metallurgical and other industries. For soft sensor, the difficult-to-measure quality variables are often estimated by those routinely measured process variables with some mathematical models [5-7]. During the last decades, soft sensor has been widely studied in the academia, and it has been extensively used for industrial process monitoring, optimization and control due to its advantages of flexibility, versatility, fast response and low cost [8-12].

Soft sensor models [13] can be divided into three categories: first-principle models (FPM) [14], data-driven models (DDM) [15] and hybrid grey models [16, 17]. FPMs are mainly based on the physicochemical mechanism of the industrial process. However, they are mostly built for the ideal steady state of the systems with certain simplified assumptions, which are difficult to model the actual production processes. Alternatively, DDMs are constructed based on the measured process data, which can describe the real process running conditions. They do not need process prior knowledge and mainly rely on the process historical data. Hence, the data quality can have a large impact on the performance of DDMs. The grey box model alleviates these problems to some extent by combining FPMs with DDMs. However, with the large-scale development of industrial processes, it becomes more and more difficult to build accurate FPMs for modern industrial processes. Differently, with the wide implementation of intelligent instrument, data collection and storage techniques, a large number of data can be collected from the industrial process, which lays a solid foundation for DDMs. Hence, DDMs have gained more and more popularity and been rapidly developed in the process industry. Moreover, DDMs have great advantages like fast response, economical efficiency and easy maintenance. Principal component regression (PCR) [18] and partial least squares regression (PLSR) [19] are typical linear data-driven methods. In order to handle nonlinear data relationships, kernel PCA [17] kernel PLS [20], support vector machine (SVM) [21] and artificial neural network (ANN) [22] have been developed for nonlinear data modeling in many industrial processes.

As a matter of fact, modern industrial processes are becoming more and more complex and large-scale with the increasing demand for production. Thus, these processes have large and varied datasets, in which there may be many unknown correlations and hidden patterns. In this way, neural networks with many hidden layers have the potentials to deal with

This paper is supported in part by the Program of National Natural Science Foundation of China (61988101, 61703440, 61590921, 61621062), in part by National Key R&D Program of China under (2018YFB1701100), and in part by the Fundamental Research Funds for the Central Universities of Central South University (2020zzts563) and supported by Hunan Provincial Innovation Foundation For Postgraduate (CX20200309).

X. Yuan, Y. Gu, Y. Wang and K. Wang are with School of Automation, Central South University, Changsha 410083, China (e-mail: yuanxf@csu.edu.cn, 1360885769@qq.com, ylwang@csu.edu.cn).

large-scale process data. However, due to the huge number of network parameters, it is often difficult and time-consuming to train the networks. Moreover, there is a risk of gradient vanishing and explosion in network training with the increasing number of hidden layers. In recent years, deep learning technique has been developed for better network training and data feature representation. For deep learning networks like deep belief network (DBN) [23] and stacked autoencoder (SAE) [24], they use the layer-wise pre-training and fine-tuning to alleviate the gradient vanishing and explosion problems. The layer-wise pre-training [25] can provide proper initial values for network parameters in the fine-tuning stage. This can reduce the possibility of gradient vanishing and explosion caused by random initializations, and then further improve the efficiency of network training. Thus, deep learning models have been extensively developed and sprung up in the past decade. Among them, DBN, which consists of multiple stacked restricted Boltzmann machines (RBM), is a hierarchical generative model that is effective for feature representation and extraction from the probabilistic view. RBMs are stochastic and generative neural networks capable of learning the internal representations of its input data. Hence, it can be applied to different tasks like image recognition. In recent years, deep neural network has been developed for effective process data analytics due to its unique advantages in tackling high-dimensional complex nonlinear quality prediction [26-28]. Especially, DBN is widely used for soft sensor in different industrial processes. For example, Liu et al. constructed DBN model on flame images for oxygen content prediction of combustion systems [29]. Shang et al. applied DBN for soft sensor estimation of the heavy diesel 95% cut point in a crude distillation unit [30]. Wang et al. developed an extended deep belief network for fault diagnosis in chemical processes [31]. To address complex prediction problems, Ji et al. proposed a new method to construct prediction intervals using deep belief networks and bootstrap [32]. Zhu and Zhang developed DBN-based soft sensors to predict the melt index of an industrial polymerization process [33]. Xibilia et al. designed a DBN to estimate the hazardous gas concentrations in industrial plants, which shows good performance with an efficient risk warning [34].

Though DBN can extract deep features from the raw input data of routine process variables, these features are learned by first unsupervised pre-training and then fine-tuning. The features can naturally represent the raw input data very well. However, the unsupervised pre-training procedure of DBN does not guarantee that the features have a good relationship with the quality variables. Instead, there may be irrelevant features with the quality variables, which are not beneficial for output prediction of the quality variables and then may deteriorate the model prediction accuracy. For soft sensor modeling, it is crucial to learn informative and useful features that are relevant with the quality variables. Meanwhile, it is important to reduce the irrelevant information from the features as much as possible. Thus, it is necessary to ensure the learned features are quality-relevant to obtain robust and accurate prediction performance in soft sensors. Hence, a better way is to introduce the quality information into the procedure of pre-training to guide the feature learning.

In order to alleviate the above problems, this paper designs a new model of supervised deep belief network (SDBN) for soft sensor, which can hierarchically learn quality-related features by stacking multiple supervised RBM (SRBM). The main contributions of this paper can be summarized as follows.

First, the quality variables are incorporated into the visible layer of RBM to construct SRBM, which can realize the learning of quality-related features from its input data in the generative model framework.

Second, SDBN is constructed to learn deep quality-related features by stacking multiple SRBMs in a hierarchical way. Since the quality information is added to each layer through hierarchical learning, the network continuously strengthens the learning of deep quality-related features that are beneficial for quality prediction.

Third, the designed method changes the pretraining of deep network into a supervised way, which can fully utilize the quality variable information for better network learning.

The remaining sections are structured as follows. Section II simply introduces the original RBM. Section III describes the proposed SRBM, SDBN, and the SDBN-based soft sensor model in detail. Section IV demonstrates the prediction performance of the SDBN model on two industrial processes. Finally, we summarize the work of this paper in Section V.

II. Restricted Boltzmann Machine (RBM)

RBM is a variant of the standard Boltzmann machine (BM) [35]. It consists of a visible layer and one hidden layer. A structure of RBM is shown in Fig. 1. Different from BM, RBM discards the connections between internal units of each layer. Only units between different layers are connected with each other. Assume v is the input visible vector and h is the hidden feature vector.

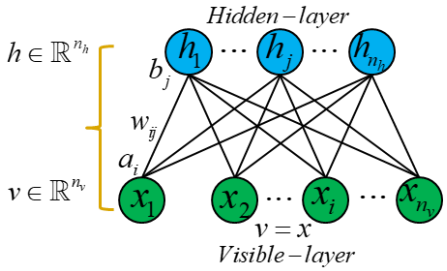


Fig.1. The structure of RBM

The visible and hidden states in the original RBM are binary values from {0, 1}. First, the energy function is defined for RBM as [25]

$$E(v,h) = -\sum_{i\in n_v, j\in n_h} w_{ij} v_i h_j - \sum_{i\in n_v} a_i v_i - \sum_{j\in n_h} b_j h_j \qquad (1)$$

where $v \in R^{n_v}$ and $h \in R^{n_h}$; n_v and n_h represent the dimension of the corresponding visible vector v and hidden vector h, respectively; v_i (i=1,2,⋯,n_v) and h_j (j = 1,2,⋯,n_h) represent unit i of the visible layer and unit j of the hidden layer, respectively; a_i and b_j are the corresponding bias terms of the

visible unit v_i and the hidden unit h_j, respectively; w_{ij} is the connection weight between v_i and h_j.

For real-valued data, the Gaussian unit-based RBM is more commonly used for energy function as [36]

$$E(v,h)=\sum_{i\in v}\frac{(a_i-v_i)^2}{2\delta_i^2}+\sum_{j\in h}\frac{(b_j-h_j)^2}{2\gamma_j^2}-\sum_{i\in v}\sum_{j\in h}\frac{v_i}{\delta_i}\frac{h_j}{\gamma_j}w_{ij} \quad (2)$$

where δ_i and γ_j are the standard deviations of the Gaussian noise of the visible unit i and the hidden unit j, respectively.

The joint probabilistic distribution of v and h can be expressed as [36]

$$P(v,h)=e^{-E(v,h)}/Z \quad (3)$$

$$Z=\sum_{v,h}e^{-E(v,h)} \quad (4)$$

where Z represents the normalized or partition function, which is a sum of the energy function over all pairs of visible and hidden vectors. Then, the marginal distributions of the visible layer and the hidden layer can be obtained as [36]

$$P(v)=\sum_{h}e^{-E(v,h)}\Big/Z \quad (5)$$

$$P(h)=\sum_{v}e^{-E(v,h)}\Big/Z \quad (6)$$

To train this RBM model, it needs to learn model parameters that can maximize the loglikelihood of the probabilistic distribution of the training visible data, which is expressed as

$$\ln(P(D))=\sum_{t=1}^{T}\ln P(v^t)=\sum_{t=1}^{T}[\ln\sum_{h}e^{-E(v^t,h)}-\ln\sum_{v,h}e^{-E(v,h)}] \quad (7)$$

where T represents the total sample number in the training dataset D, in which $D=\{v^1,v^2,....,v^T\}$, $t=1,2,...,T$. Here, v^t represents the t-th training sample, and $v^t=(v_1^t,v_2^t,...,v_{n_v}^t)$.

For a representative sample v^t, the gradient of $\ln P(v^t)$ with regard to the network parameters can be calculated as follows [36]

$$\frac{\partial\ln P(v^t)}{\partial\theta}=-\sum_{h}P(h\mid v^t)\frac{\partial E(v^t,h)}{\partial\theta}+\sum_{v,h}P(v,h)\frac{\partial E(v,h)}{\partial\theta} \quad (8)$$

where θ is the network parameter set with $\theta=\{w_{ij},a_i,b_j\}$.

The two terms in the right of the formula are the expectations of the gradient function under distribution $P(h\mid v^t)$ and distribution $P(v,h)$, respectively. However, the second term involves the joint distribution of v and h, which is often difficult to obtain. Thus, the contrast divergence (CD) [37] algorithm proposed by Hinton et al. is often adopted to estimate the joint distribution. For DBN with multiple RBMs, it can be built using the layer-wise greedy pre-training through the above procedure for each RBM.

III. Supervised DBN

In actual industrial processes, most of the collected data is continuous real value. Therefore, it is suitable to use Gaussian unit-based RBM for data-driven soft sensor modeling. Though the original DBN can learn deep features through pre-training, it cannot learn the quality-related information from the industrial process data. For soft sensor modeling, it is important to learn informative features that are related to the quality variable in order to ensure the prediction accuracy. To deal with this problem, a supervised restricted Boltzmann machine (SRBM) is first designed to learn quality-related features by integrating the quality variable with the process variables to the visible layer. Then, a supervised deep belief network (SDBN) is constructed to learn deep quality-relevant features by stacking multiple SRBMs hierarchically.

A. SRBM

Assume $x=[x_1,\cdots x_i,\cdots x_{n_x}]^{\mathrm{T}}\in R^{n_x}$ ($i=1,2,...,n_x$) is the raw input vector and $y=[y_1,\cdots y_l,\cdots y_{n_y}]^T\in R^{n_y}$ ($l=1,2,...,n_y$) is the quality vector. The structure of the SRBM is shown in Fig. 2. It can be seen from Fig. 2 that both the input variable vector x and the quality variable vector y are introduced into the visible layer in SRBM to simultaneously learn the hidden layer features. By integrating the quality variable y with process variables x into the visible layer v, the network takes the combined vector $v=(x,y)\in R^{n_v+n_y}$ as a new visible variable vector. In this way, quality-related features can be learned in the hidden layer with the guidance of quality variables.

After the quality information is introduced, the energy function for the new input variable $v=(x,y)$ of the SRBM can be expressed as

$$E(x,y,h)=\sum_{i\in n_v}\frac{(a_i-x_i)^2}{2\sigma_i^2}+\sum_{l\in n_y}\frac{(c_l-y_l)^2}{2\beta_l^2}+\sum_{j\in n_h}\frac{(b_j-h_j)^2}{2\gamma_j^2}-\sum_{i\in n_v}\sum_{j\in n_h}\frac{x_i}{\sigma_i}\frac{h_j}{\gamma_j}w_{ij}-\sum_{l\in n_y}\sum_{j\in n_h}\frac{y_l}{\beta_l}\frac{h_j}{\gamma_j}w_{lj} \quad (9)$$

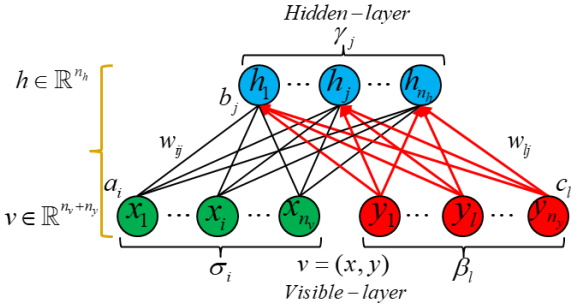


Fig.2. The detailed structure of the SRBM

where a_i, c_l and b_j are the bias terms of the input variable unit i, quality variable unit l and hidden unit j, respectively; σ_i, β_l and γ_j are the corresponding standard deviations of the Gaussian noise of the input variable unit i, quality variable unit l and hidden unit j, respectively. w_{ij} is the connection weight between input variable unit x_i and the hidden layer unit h_j. w_{lj} is the connection weight between quality variable unit y_l and the hidden layer unit h_j.

After calculating the energy function, we can get the joint distribution and marginal distribution as follows

$$P(x,y,h)=e^{-E(x,y,h)}\Big/\int_{(x,y,h)}e^{-E(x,y,h)} \tag{10}$$

$$P(x,y)=\int_h P(x,y,h)=\int_h e^{-E(x,y,h)}-\int_{(x,y,h)}e^{-E(x,y,h)} \tag{11}$$

Given the training dataset $\{(x^1,y^1),\cdots,(x^T,y^T)\}$, it is necessary to maximize the log-likelihood function of the training data to learn the SRBM network parameters, which is denoted as $\theta=\{w_{ij},w_{lj},a_i,c_l,b_j,\sigma_i,\beta_l,\gamma_j\}$. Here, T is still used to represent the number of training samples. Also, each training sample with a pair of input and quality parts can be denoted as $v^t=(x^t,y^t),\ t=1,2,...,T$. Then, the log-likelihood function of the training dataset is

$$\ln(\prod_{t=1}^{T}P(v^t))=\sum_{t=1}^{T}[\ln\int_h e^{-E(x^t,y^t,h)}-\ln\int_{(x,y,h)}e^{-E(x,y,h)}] \tag{12}$$

Also, it is necessary to compute the gradient of the log-likelihood function with regard to each of the model parameters in order to maximize it. For a given sample $v^t=(x^t,y^t)$, the gradient terms for the SRBM with regard to parameter $\theta=\{w_{ij},w_{lj},a_i,c_l,b_j,\sigma_i,\beta_l,\gamma_j\}$ are calculated as

$$\frac{\partial\ln(P(v^t))}{\partial w_{ij}}=<\frac{x_i^t}{\sigma_i}\frac{h_j}{\gamma_j}>_{P(h|v^t)}-<\frac{x_i}{\sigma_i}\frac{h_j}{\gamma_j}>_{P(v,h)} \tag{13}$$

$$\frac{\partial\ln(P(v^t))}{\partial w_{lj}}=<\frac{y_l^t}{\beta_l}\frac{h_j}{\gamma_j}>_{P(h|v^t)}-<\frac{y_l}{\beta_l}\frac{h_j}{\gamma_j}>_{P(v,h)} \tag{14}$$

$$\frac{\partial\ln(P(v^t))}{\partial a_i}=<\frac{x_i^t}{\sigma_i^2}>_{P(h|v^t)}-<\frac{x_i}{\sigma_i^2}>_{P(v,h)} \tag{15}$$

$$\frac{\partial\ln(P(v^t))}{\partial c_l}=<\frac{y_l^t}{\beta_l^2}>_{P(h|v^t)}-<\frac{y_l}{\beta_l^2}>_{P(v,h)} \tag{16}$$

$$\frac{\partial\ln(P(v^t))}{\partial b_j}=<\frac{h_j}{\gamma_j^2}>_{P(h|v^t)}-<\frac{h_j}{\gamma_j^2}>_{P(v,h)} \tag{17}$$

$$\begin{aligned}\frac{\partial\ln(P(v^t))}{\partial\sigma_i}=&<\frac{-(a_i-x_i^t)^2}{\sigma_i^3}+\sum_j\frac{x_i^t h_j w_{ij}}{\sigma_i^2\gamma_j}>_{P(h|v^t)}\\&-<\frac{-(a_i-x_i)^2}{\sigma_i^3}+\sum_j\frac{x_i h_j w_{ij}}{\sigma_i^2\gamma_j}>_{P(v,h)}\end{aligned} \tag{18}$$

$$\begin{aligned}\frac{\partial\ln(P(v^t))}{\partial\beta_l}=&<\frac{-(c_l-y_l^t)^2}{\beta_l^3}+\sum_j\frac{y_l^t h_j w_{lj}}{\beta_l^2\gamma_j}>_{P(h|v^t)}\\&-<\frac{-(c_l-y_l)^2}{\beta_l^3}+\sum_j\frac{y_l h_j w_{lj}}{\beta_l^2\gamma_j}>_{P(v,h)}\end{aligned} \tag{19}$$

$$\begin{aligned}\frac{\partial\ln(P(v^t))}{\partial\gamma_j}=&<\frac{-(b_j-h_j)^2}{\gamma_j^3}+\sum_i\frac{x_i^t h_j w_{ij}}{\gamma_j^2\sigma_i}+\sum_l\frac{y_l^t h_j w_{lj}}{\gamma_j^2\beta_l}>_{P(h|v^t)}\\&-<\frac{-(b_j-h_j)^2}{\gamma_j^3}+\sum_i\frac{x_i h_j w_{ij}}{\gamma_j^2\sigma_i}+\sum_l\frac{y_l h_j w_{lj}}{\gamma_j^2\beta_l}>_{P(v,h)}\end{aligned} \tag{20}$$

where $<formula>_p$ indicates the mathematical expectation of $formula$ over distribution p in the subscript.

Since it is often very difficult to directly calculate the second term in equations (13)-(20), the contrast divergence (CD) algorithm is adopted for approximate calculation. The gradient terms can be eventually calculated as shown in APPENDIX.

Through the above procedure, gradient descend algorithm can be carried out iteratively to learn the network parameter $\theta=\{w_{ij},w_{lj},a_i,c_l,b_j,\sigma_i,\beta_l,\gamma_j\}$ for SRBM with all the training data.

B. Supervised deep belief network

The basic SRBM modules can be hierarchically stacked to construct supervised deep belief network (SDBN). An illustrative structure of SDBN is shown in Fig. 3. SDBN consists of multiple SRBMs and an additional output layer. Assume there are a total of *S* SRBM modules in SDBN. The *s*th SRBM is denoted as $\text{SRBM}_{[s]}$. Here, subscript [5] is used to identify the terms with regard to the *s*th SRBM. The training of SDBN consists of the pre-training and fine-tuning stages.

Pre-training:

First, the original input and quality variable is combined as the visible state $v_{[1]}^t=(x^t,y^t),t=1,2,...,T$ to serve as the input for the first SRBM ($\text{SRBM}_{[1]}$). By carrying out iterative back-propagation (BP) based gradient descend algorithm, $\text{SRBM}_{[1]}$ can be trained with the training data $\{(x^1,y^1),\cdots,(x^T,y^T)\}$. Moreover, the first-layer hidden feature state can be obtained as $\left\{h_{[1]}^1,h_{[1]}^2,...,h_{[1]}^T\right\}$ by forward propagation from the training data. Then, the first-layer hidden feature data $h_{[1]}^t,\ t=1,2,...,T$ is further combined with the quality data $y^t,t=1,2,...,T$ to serve as the visible input $v_{[2]}^t=(h_{[1]}^t,y^t),\ t=1,2,...,T$ for the second SRBM ($\text{SRBM}_{[2]}$) to learn the second-layer feature data $h_{[2]}^t,\ t=1,2,...,T$. Similarly, BP algorithm is carried out to train $\text{SRBM}_{[2]}$ on the new training data $\{(h_{[1]}^1,y^1),\cdots,(h_{[1]}^T,y^T)\}$. In the similar way, if $SRBM_{[s]}$ is pretrained, the *s*th-layer feature data can be obtained as $\left\{h_{[s]}^1,h_{[s]}^2,...,h_{[s]}^T\right\}$. Then, $h_{[s]}^t,\ t=1,2,...,T$ will be combined with the quality data $y^t,t=1,2,...,T$ as the new input $v_{[s+1]}^t=(h_{[s]}^t,y^t),t=1,2,...,T$ for $SRBM_{[s+1]}$ to learn higher-layer features by pre-training. This procedure is carried out until the last SRBM ($SRBM_{[S]}$) is pretrained on its training data $\{(h_{[S-1]}^1,y^1),\cdots,(h_{[S-1]}^T,y^T)\}$.

From the above procedure, it can be seen that the pre-training of SDBN starts from $SRBM_{[1]}$ to $SRBM_{[S]}$ in a layer-by-layer greedy manner. Thus, it can learn deep hierarchical features for process data. Moreover, SDBN can keep the deep features largely quality-relevant since the quality variable vector is additionally added to each input layer of SRBM in a stacked way.

Fine-tuning:

After pre-training, an output layer is added to the top hidden layer for quality prediction as shown in blue circles in Fig. 3. The pre-training allows SDBN to find appropriate initial weights for better and faster modeling in subsequent

fine-tuning step. In the fine-tuning step, the weight and bias parameters from the input to the last hidden layer of the SDBN network is first initialized with the network parameters obtained in the pre-training step. Moreover, the parameters between the output layer and the last hidden layer are randomly initialized. Then, the predicted quality variable is calculated at the output layer as

$$\hat{y}^t = affine(w_{[o]}h^t_{[S]} + b_o) \tag{21}$$

where *affine*(•) is the activation function of the output layer. $w_{[o]}$ and b_o are the connection weight and bias from the last hidden layer to the output layer. $\hat{y}^t$ is the predicted output value for sample *t*.

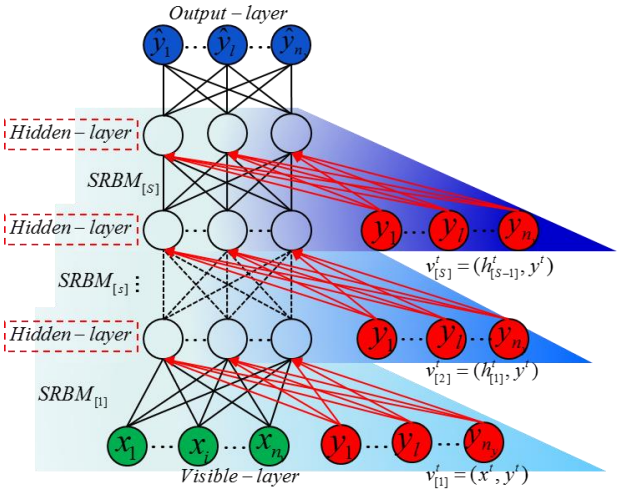


Fig. 3. The detailed structure of the SDBN

In order to fine tune the SDBN network, the root mean squares error (RMSE) index is calculated for the training data as

$$RMSE_{tr} = \sqrt{\sum_{t=1}^{T} \left\| y^t - \hat{y}^t \right\|^2 / T_{tr}} \tag{22}$$

where T_{tr} is the total number of samples of the training dataset.

After calculating the RMSE of the network, the back-propagation algorithm is utilized to fine tune the parameters of the whole network until some convergence conditions are met.

Testing:

After SDBN is trained offline, it can be used for online prediction of testing samples. For the *K*th sample in the testing stage, its raw input data part can be denoted as $x^{T_{tr}+K}$. As for SDBN, its network inputs should contain both the raw input and quality part in order to carry out forward propagation for output prediction. Different from the training stage, in which the quality variable information is already available for the training samples, the quality variables are unknown and to be predicted for the testing samples. Therefore, it is necessary to give an initial rough estimation for the quality variables of each testing sample. The rough estimation can be achieved by adopting some basic models like PCR. Considering the temporal correlation of process industries, we use the quality information obtained at the previous moment as the rough quality estimation of the current testing sample. That is to say, we make an initial guess based on $y^{T_{tr}+K-1}$ for the *K*th testing sample as

$$\hat{y}^{T_{tr}+K}_{\text{initial}} = y^{T_{tr}+K-1} \tag{23}$$

It is worth noting that when $y^{T_{tr}+K-1}$ is unavailable, the initial guess for the *K*th testing sample can be first estimated by the predicted output of SDBN of the (*K*-1)th sample as

$$\hat{y}^{T_{tr}+K}_{\text{initial}} = \hat{y}^{T_{tr}+K-1} \tag{24}$$

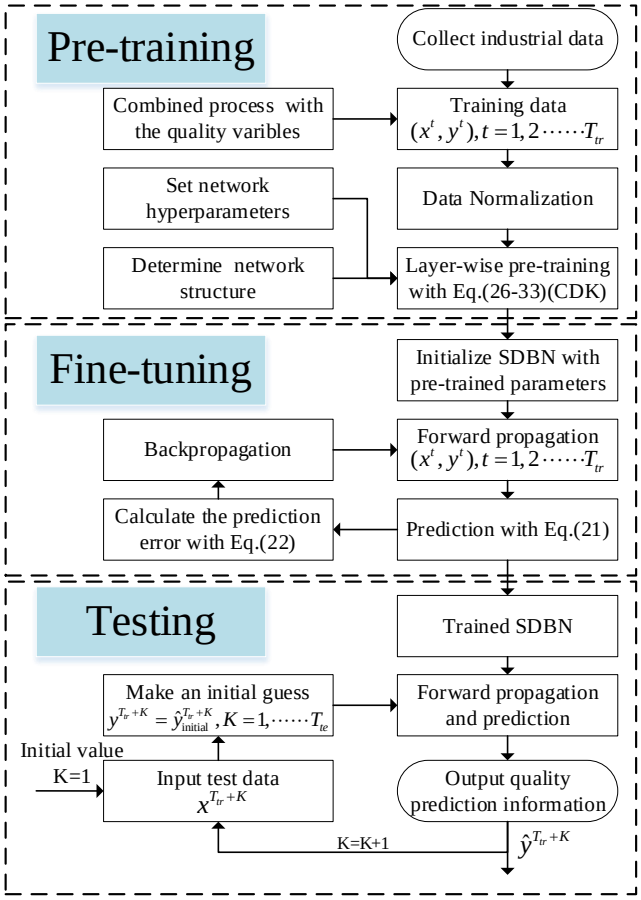


Fig.4. The flowchart of the proposed SDBN-based soft sensor model

Then, the raw input variable part and the initial estimation part of quality variable part can be combined as the visible input data for the SDBN for quality prediction. The visible input data for the *K*th testing sample is denoted as $v^{T_{tr}+K} = (x^{T_{tr}+K}, \hat{y}^{T_{tr}+K}_{\text{initial}})$.

Finally, by substituting $v^{T_{tr}+K}$ into the trained SDBN model, the final predicted quality value can be obtained as $\hat{y}^{T_{tr}+K}$ by forward propagation from the first to the last layer of SDBN.

Fig. 4 shows the main procedures of the SDBN-based soft sensor. After the prediction is finished for the testing dataset, its RMSE is calculated as

$$RMSE_{te} = \sqrt{\sum_{K=1}^{T_{te}} \left\| y^{T_{tr}+K} - \hat{y}^{T_{tr}+K} \right\|^2 / T_{te}} \tag{25}$$

where T_{te} is the total number of samples of the test dataset.

IV. CASE STUDIES

In this section, the performance of the proposed SDBN-based soft sensor model is evaluated on an industrial debutanizer column and an actual hydrocracking process. The

configurations of the simulation computer are as follows: Operation System: Windows 10; CPU: Intel i5-8400 (2.80GHz); RAM: 8GB; Software: Python 3.6. In the two cases, in order to ensure the continuity and consistency of data distribution, and the generalization of the model, the data set is divided into training set and test set at a ratio of about 2 : 1 [38].

A. *Debutanizer column*

The debutanizer column [39] is an important gas recovery unit of the desulfuring and naphtha splitter plant in the refinery process. The goal is to produce liquefied petroleum gas as its top product and light naphtha as its bottom product from naphtha gasoline stream. The studied process is located at the ERG Raffineria Mediterranea s.r.l in Syracuse of Italy. A flowchart of the debutanizer column is shown in Fig. 5.

The process mainly consists of six devices including the heat exchanger, condenser, reboiler, reflux pump, splitter and reflux accumulator. Usually, the unstabilized naphtha is fed into this process, which contains ingredients of different alkanes from the crude distiller. Then, the naphtha is boiled and fractionated with high temperatures and pressure differentials in the distillation column. The lights ends are removed from the top and the debutanized naphtha is removed from the bottom, which is directly sent to the splitter section for further processing. The condenser in the overhead section condenses the overhead vapor into liquid to the liquefied petroleum gas section. Meanwhile, a part of the condensed liquid petroleum from the overhead is used as a reflux to the column. Moreover, the reboiler provides the heat that is necessary to partially vaporize the liquid before returning it to the column.

To ensure stable production and maintain the product quality at a desired level, it is necessary to identify the top and bottom compositions of the debutanizer column quickly with a high degree of precision. In the bottom, an objective is to minimize the content of butane (C4) as much as possible in order to improve the control quality. Hence, it is critically important to carry out real-time measurement of the butane content to improve the control performance and ensure product quality. However, in the actual production, it is difficult to directly measure the butane concentration online on the bottom flow. Instead, the C4 content is measured on the overhead of the deisopentanizer column [39] after the debutanizer by gas chromatography, which causes a large measurement lag due to the distant chromatograph location and analyzing cycle. Therefore, it is necessary to realize the online estimation of butane concentration by establishing soft sensor models. In order to accurately predict the butane concentration, seven easily measurable auxiliary variables that are most related to the butane concentration in mechanism are selected as the inputs of the soft sensor model. The measurement locations of the seven variables are indicated in the circles (U1-U7) in Fig. 5. The detailed description of the seven variables is shown in Table I.

2394 samples were collected from the industrial site [39]. To ensure the generalization of the model [38], the first 1500 samples are used as the training dataset and the rest 894 ones are as the testing dataset. As can be seen, the dimensions of the raw input data x^t and the quality data y^t are 7 and 1, respectively. Then, SDBN-based soft sensor model is constructed and trained with the training data for quality prediction.

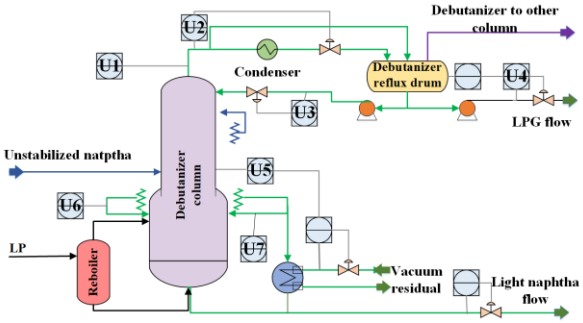


Fig. 5. Flowchart for the debutanizer column [39]

TABLE I. Secondary variables for the soft sensor on the debutanizer column

Secondary variables	Description
U1	Top temperature
U2	Top pressure
U3	Flow of reflux
U4	Flow to the next process
U5	Temperature of the sixth tray
U6	Temperature A at bottom
U7	Temperature B at bottom

TABLE II. Performance comparison of SDBN model with different learning rates on the debutanizer column

Learning Rate	1e-2	1e-3	1e-4
Training RMSE	0.0025	**0.0004**	0.0015
Testing RMSE	0.0074	**0.0007**	0.0263

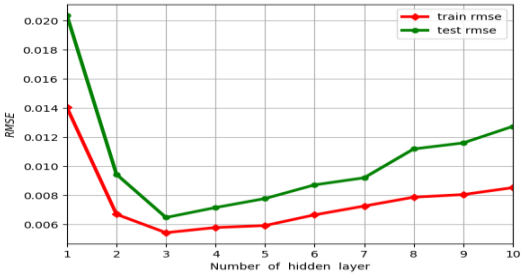


Fig. 6. Comparison of RMSE with different number of hidden layers for SDBN on the debutanizer column

For some of the commonly used hyper-parameters of deep networks, a practical guide with recommendations are introduced by Bengio in Ref. [40]. In this paper, for the hyperparameter of epochs, it is set as 30 and 100 in the pretraining and fine-tuning stages, respectively. While for the hyperparameter of learning rate, it is set as 1e-3 and 1e-4 in the pre-training and fine-tuning stages, respectively.

Taking the learning rate in the fine-tuning procedure for example, it has a very important impact on the learning effect and the reliability of the model. In order to choose an appropriate learning rate, simulation experiments are conducted for the model with different learning rates. Table II shows the detailed results of SDBN with different learning rates in the fine-tuning. It can be seen that in the fine-tuning process, when the learning rate is 1e-2, the large learning rate causes the

model to fail to learn the best sample representation during training. When the learning rate is 1e-4, the model may be over-fitting, resulting in poor model performance. If the learning rate is 1e-3, the model has better performance on both training and testing datasets. Therefore, the learning rate of the is set to 1e-3 in the fine-tuning procedure of SDBN model.

Also, it is rather difficult to find the optimal network structure. There is no scientific way with good principle for obtaining an optimal deep network structure. Generally, the trial and error technique is often used by developing a number of neural networks with different structures and then obtain the final structure by comparing their performance. In order to find the proper network structure in this study, several SDBN networks are built with different number of hidden layers. The number of hidden layers is selected from range set [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. The RMSE index is used to evaluate the prediction performance of the SDBN models. The results are shown in Fig. 6 for the RMSE values on the training and testing datasets with regard to different number of hidden layers. From Fig. 6, it is easily seen that both the training dataset and the testing dataset have the smallest RMSE values when the number of hidden layers is 3. Thus, the network with three hidden layers is used as the optimal model structure in this case study. Moreover, the numbers of neurons are 40, 40, 20 in the first, second and third hidden layer of the developed network, respectively.

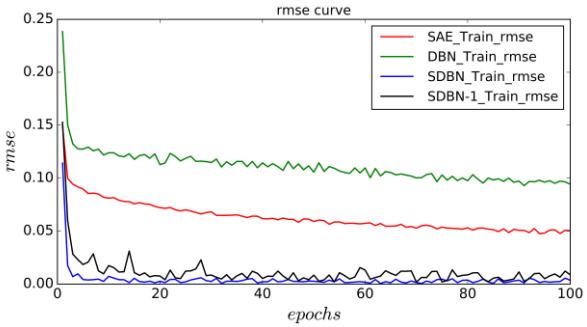


Fig.7. Fine-tuning procedure of the four methods on the debutanizer column

TABLE III. Training and testing RMSEs of the four methods on the debutanizer column example

Method	Training RMSE	Testing RMSE
SAE	0.0974	0.1776
DBN	0.0759	0.1827
SDBN-1	0.0023	0.0119
SDBN	0.0004	0.0007

In order to demonstrate the performance of the SDBN network, three other soft sensor models are constructed based on SAE, DBN, and DBN with quality variable added only at the first layer. The third method is denoted as SDBN-1. In a similar way, these networks can be trained and used for prediction of the testing dataset. First, Fig. 7 shows the RMSE trend with the iteration in the fine-tuning procedure after pre-training for the four methods. Here, the number of iterations is set to 100 for fine-tuning. As can be seen, SAE and DBN have a fine-tuning convergent state in the range of [0.05, 0.15]. Differently, SDBN-1 and SDBN can achieve convergence with fine-tuning loss much smaller than 0.05. After the networks are trained, Table III show the RMSE values on the training and testing datasets for the four models. It can be seen that the RMSEs of the SAE model and DBN model are much higher than the results of SDBN-1 and SDBN. This is because SAE and DBN extract deep features from the original data by unsupervised pre-training. These features may contain much irrelevant information with the quality variables. By introducing the quality variable at the first input layer of DBN, SDBN-1 can largely improve the prediction performance. However, SDBN introduces the quality information *y* in each visible layer during the pre-training, thereby guides the network to learn deep features that are related to the quality variable. The quality-relevant information can be progressively enhanced from low to high layers. In this way, SDBN can predict the quality variable the best.

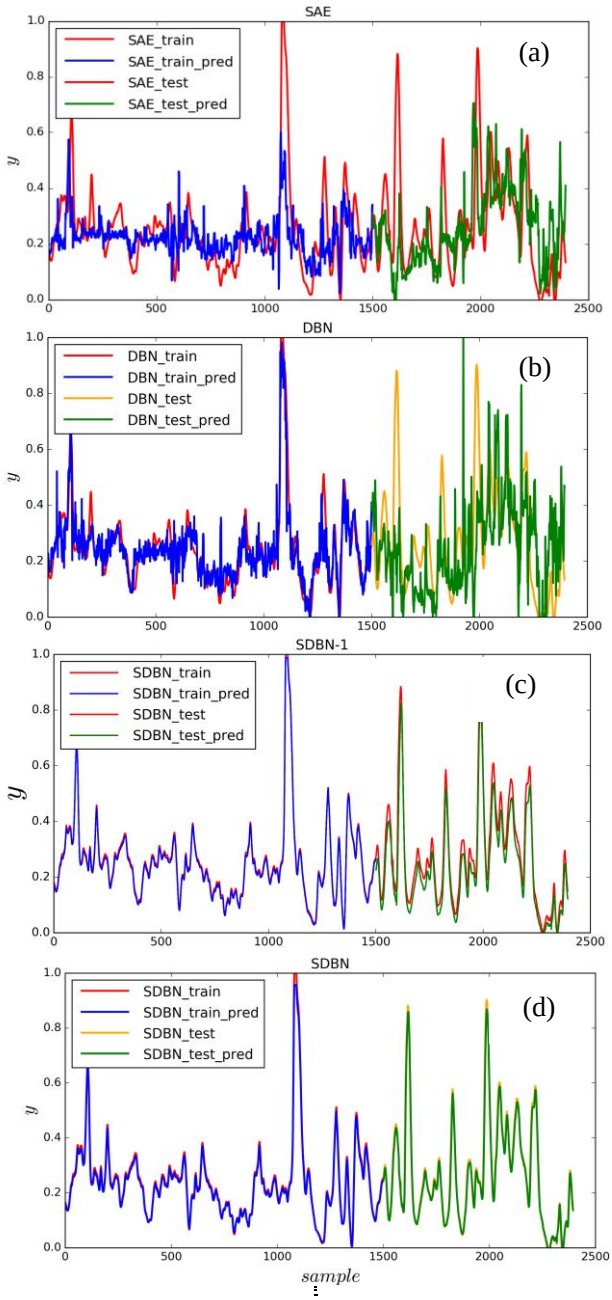


Fig.8. Detailed Predictions of the four methods for the training and testing datasets on the debutanizer column

At last, Fig. 8 further shows the detailed prediction results on

both the training and testing data samples with the four methods. It can be seen that the predicted values of SDBN track best with the real values on both datasets. For SDBN-1, it can track with the trend of the real output curve and still has deviations for some testing points. However, the predicted curves of SAE and DBN have relatively large errors with the real quality variable curve. The results show that the prediction performance of SDBN can be largely improved by introducing the quality variables into the network layer by layer.

B. Hydrocracking process

Hydrocracking [41] is one of the most important techniques in petroleum refining processes. The purpose of hydrocracking process is to transform the macromolecules into small molecules to improve the yield of light oils, which are generally converted into light oils like naphtha, kerosene and diesel. The hydrocracking process studied in this paper is from an industrial refining company in China. Fig. 9 shows a flowchart of this hydrocracking process. Usually, online monitoring of key product qualities is required to ensure production safety and product yield in this process. One of the most important quality indicators is the C4 content in naphtha, which reflects the quality of naphtha product. However, this indicator can only be measured offline by lab test. For this continuous production process, it causes large time delay for real-time process monitoring and optimization. Therefore, it is very beneficial to realize online prediction of C4 content using the easy-to-measure process variables by soft sensor modeling for real-time monitoring and online optimization of the process.

Through mechanism analysis, 43 easy-to-measure process variables, like the flow-rate, temperature and pressure, are selected as the secondary variables for soft sensor modeling throughout the production process from inlet to outlet. These secondary variables have highly complex nonlinear relationships with the quality variable of C4 content. Detailed description of these variables can be found in Ref. [42]. Therefore, deep learning techniques are adopted for soft sensing of the quality variable. Here, the four models of SAE, DBN, SDBN-1 and SDBN are constructed to predict the C4 content in the naphtha. A total of 868 data samples were collected during the production process. The first 600 were used for training and the remaining 268 for testing according to the data sampling sequence. As can be seen, the dimension of the raw input data x^t and quality label y^t are 43 and 1, respectively.

The hyperparameters of the epoch and learning rate are the same as those in the debutanizer column. Again, the learning rate in the fine-tuning procedure is taken for example to investigate the network parameter sensitivity for approximate value. With different values of learning rate, the performance of SDBN model is shown in Table IV for the training and testing dataset. It can be seen from Table IV, the model can also achieve the best performance when the learning rate is set to 1e-3. Also, the network structure of SDBN is investigated for the prediction performance in this process. The number of hidden layers is also selected from range set [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], respectively. Fig. 10 shows the model prediction RMSE with respect to the number of hidden layers. It can be seen from Fig. 10 that when the number of hidden layers increases from 1 to 5, the prediction performance gets better for both the training and testing datasets. If the number of hidden layers exceeds 5, the RMSE fluctuates only in a small range and become relatively stable. In this way, the number of hidden layers is set as 5 for the SDBN network. In addition, the numbers of neurons are determined as 128, 64, 64, 32, 32 for the five hidden layers, respectively.

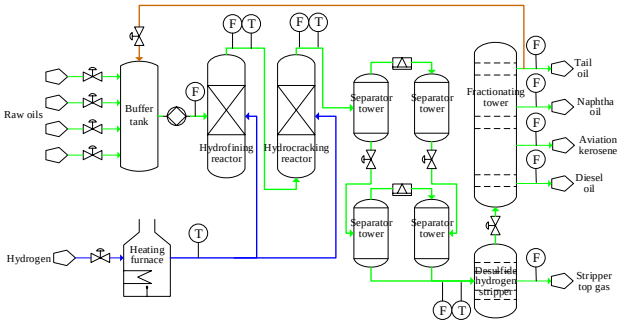


Fig.9. Flowchart of the Hydrocracking process

TABLE IV. Performance comparison of SDBN model with different learning rates on the hydrocracking process

Learning Rate	1e-2	1e-3	1e-4
Training RMSE	0.0106	**0.0007**	0.0014
Testing RMSE	0.0111	**0.0017**	0.0021

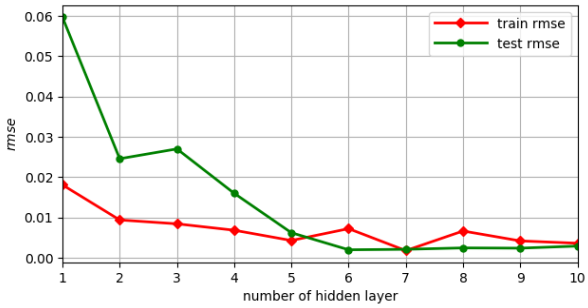


Fig. 10 Comparison of RMSE with different number of hidden layers for SDBN on the hydrocracking process

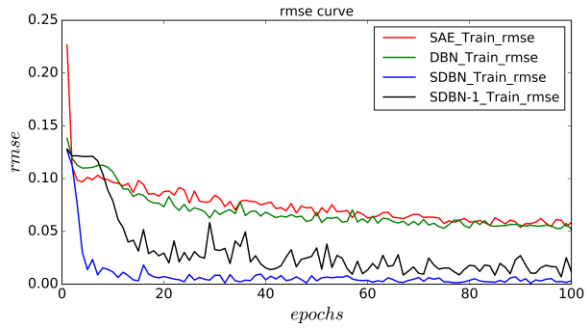


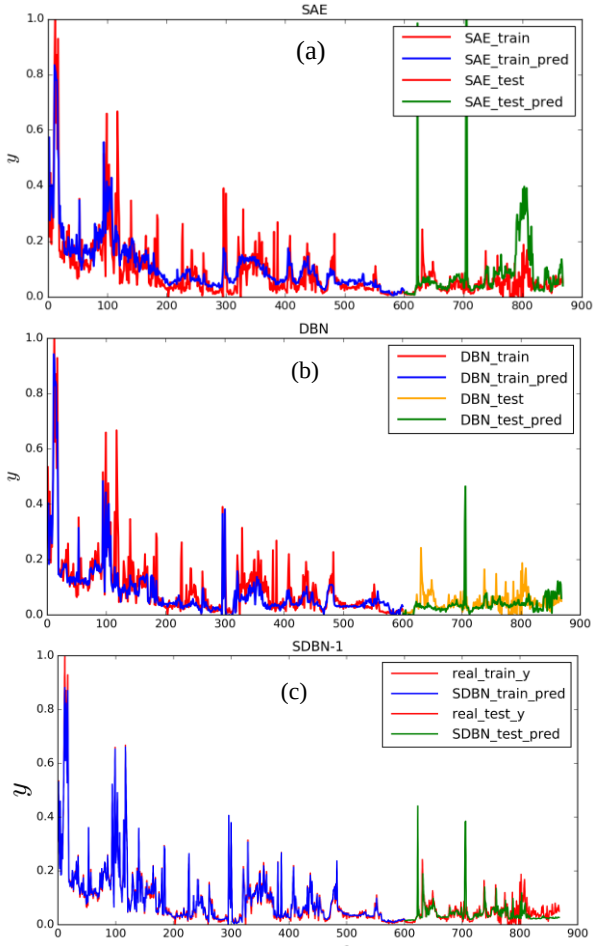
Fig.11. Fine-tuning procedure of the four models on the hydrocracking process

TABLE V. Training and testing RMSEs of the four method on the hydrocracking process

Method	Training RMSE	Testing RMSE
SAE	0.0432	0.0832
DBN	0.0493	0.0327
SDBN-1	0.0086	0.0192
SDBN	0.0007	0.0017

Here, the four methods are constructed and compared for prediction performance of this hydrocracking process. First,

Fig. 11 shows the RMSE trend with regard to the iteration in the fine-tuning procedure for the four models. Here, the number of iterations is still 100 for the fine-tuning procedure. As can be seen, SDBN has a smaller convergent state than SAE, DBN and SDBN-1. Table V gives the detailed prediction RMSE on both the training and testing datasets. The RMSE values of the SAE and DBN models are much larger than that of SDBN-1 and SDBN. Based on the original DBN, SDBN-1 and SDBN introduce the quality information into the training process for quality-related feature learning, which largely improves the prediction accuracy. Moreover, since SDBN utilize the quality variables to enhance the learning direction in a layer-wise way, it can further improve the prediction performance than SDBN-1. Compared to the other three methods, the RMSE of the SDBN network is reduced to 0.0007 and 0.0017 on the training and testing datasets, respectively. Moreover, Fig. 12 shows the detailed predictions on the training and testing samples for the four deep networks. It can be seen from Fig. 12 that the predicted values of SDBN-based soft sensor can track much better with the actual output values than these of SAE, DBN and SDBN-1 on both the training and testing datasets.



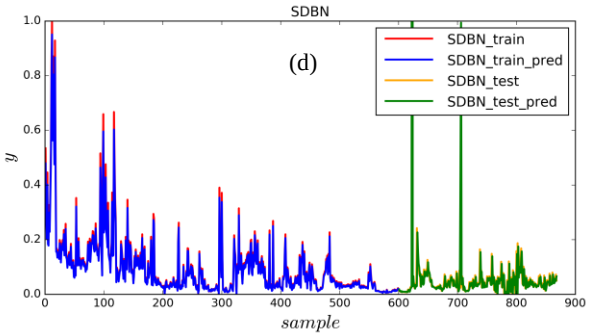


Fig.12. Predictions for the hydrocracking process: (a) SAE; (b) DBN; (c) SDBN.

Based on the analysis and simulations in this paper, comparison is provided to illustrate the advantages and disadvantages of SDBN and some other existing technologies for quality prediction of industrial processes from four aspects that are commonly used in evaluation models. Table VI provides the comparison of SDBN with DBN, SAE, PCR and PLS. It can be seen from Table VI that the SDBN model proposed in this study has the same advantages over the deep learning models such as DBN and SAE in the first two indicators. At the same time, the simulation results in this paper shows that the SDBN model is more accurate than the other models. The shortcoming is that SDBN is a static model and does not have dynamic learning ability for time series.

Table VI: Comparison of model advantages and disadvantages

Method	Nonlinearity (√/× =Nonlinear /Linear)	Data representation capacity (√/○/× =High /General /Low)	Precision (√/○/× =Precision /General /Imprecision)	Dynamics (√/× =Dynamic /static)
SDBN	√	√	√	×
DBN	√	√	○	×
SAE	√	√	○	×
PCR	×	○	○	×
PLSR	×	○	○	×

V. Conclusion

This paper mainly focuses on the application of soft sensor in the process industry like industrial metallurgical, chemical and refining processes. Especially, soft sensors are developed for key quality variable prediction of two plants of industrial refining process with SDBN. First, a supervised restricted Boltzmann machine (SRBM) is proposed for quality-related feature learning for process data, in which the quality variable is added to the visible layer of basic RBM. With this basic module, a supervised deep belief network is further established by stacking multiple SRBMs progressively. Thus, SDBN can learn deep quality-relevant features for soft sensor modeling. The SDBN enriches the development of deep learning-based virtual measurement technology in the field of instrument and measurement. Apart from the two cases on a debutanizer column and a hydrocracking process, the proposed method is also used for soft sensor applications of an industrial sintering

process in the ironmaking industry, which can successfully predict the quality attributes of sinter like the contents of ferrous oxide, total Fe and basicity. Moreover, we have developed software tool for quality prediction of an industrial sintering process in China. By far, the software tool developed based on the SDBN model has been applied to the industrial sintering process in the ironmaking industry for about a year, which greatly improves the control performance of the process and ensures the product quality. The main practical issue of implementation of this soft sensor in real-time environment is how to deal with the time-varying process conditions in order to keep the high model performance. Hence, one research direction is to build adaptive SDBN model for real-life industry applications. For example, online fine-tuning strategy can be applied for adaptive modeling to deal with the changing working conditions of industrial processes. Moreover, just like DBN, SDBN is still a static model that is difficult to modeling the dynamics of process time series. A potential future work may focus on the development of dynamic modeling strategies based on DBN and SDBN to improve model representation and prediction performance.

APPENDIX: Detailed gradient calculation formulas

This Appendix presents the gradient calculation formula derived from formula (13)-(20) as

$$\frac{\partial \ln(P(v^t))}{\partial w_{ij}} = \Delta w_{ij} \approx \frac{x_i^{t(0)}}{\sigma_i}\frac{h_j^{(0)}}{\gamma_j} - \frac{x_i^{t(k)}}{\sigma_i}\frac{h_j^{(k)}}{\gamma_j} \tag{26}$$

$$\frac{\partial \ln(P(v^t))}{\partial w_{lj}} = \Delta w_{lj} \approx \frac{y_l^{t(0)}}{\beta_l}\frac{h_j^{(0)}}{\gamma_j} - \frac{y_l^{t(k)}}{\beta_l}\frac{h_j^{(k)}}{\gamma_j} \tag{27}$$

$$\frac{\partial \ln(P(v^t))}{\partial a_i} = \Delta a_i \approx \frac{x_i^{t(0)}}{\sigma_i^2} - \frac{x_i^{t(k)}}{\sigma_i^2} \tag{28}$$

$$\frac{\partial \ln(P(v^t))}{\partial c_l} = \Delta c_l \approx \frac{y_l^{t(0)}}{\beta_l^2} - \frac{y_l^{t(k)}}{\beta_l^2} \tag{29}$$

$$\frac{\partial \ln(P(v^t))}{\partial b_j} = \Delta b_j \approx \frac{h_j^{(0)}}{\gamma_j^2} - \frac{h_j^{(k)}}{\gamma_j^2} \tag{30}$$

$$\begin{aligned}\frac{\partial \ln(P(v^t))}{\partial \sigma_i} = \Delta\sigma_i \approx &< \frac{-(a_i - x_i^{t(0)})^2}{\sigma_i^3} + \sum_j \frac{x_i^{t(0)} h_j^{(0)} w_{ij}}{\sigma_i \gamma_j} > \\ &- < \frac{-(a_i - x_i^{t(k)})^2}{\sigma_i^3} + \sum_j \frac{x_i^{t(k)} h_j^{(k)} w_{ij}}{\sigma_i \gamma_j} >\end{aligned} \tag{31}$$

$$\begin{aligned}\frac{\partial \ln(P(v^t))}{\partial \beta_l} = \Delta\beta_l \approx &< \frac{-(c_l - y_l^{t(0)})^2}{\beta_l^3} + \sum_j \frac{y_l^{t(0)} h_j^{(0)} w_{lj}}{\beta_l \gamma_j} > \\ &- < \frac{-(c_l - y_l^{t(k)})^2}{\beta_l^3} + \sum_j \frac{y_l^{t(k)} h_j^{(k)} w_{lj}}{\beta_l \gamma_j} >\end{aligned} \tag{32}$$

$$\begin{aligned}\frac{\partial \ln(P(v^t))}{\partial \gamma_j} &= \Delta\gamma_j \\ \approx &< \frac{-(b_j - h_j^{(0)})^2}{\gamma_j^3} + \sum_i \frac{x_i^{t(0)} h_j^{(0)} w_{ij}}{\gamma_j^2 \sigma_i} + \sum_l \frac{y_l^{t(0)} h_j^{(0)} w_{lj}}{\gamma_j^2 \beta_l} > \\ - < &\frac{-(b_j - h_j^{(k)})^2}{\gamma_j^3} + \sum_i \frac{x_i^{(k)} h_j^{(k)} w_{ij}}{\gamma_j^2 \sigma_i} + \sum_l \frac{y_l^{(k)} h_j^{(k)} w_{lj}}{\gamma_j^2 \beta_l} >\end{aligned} \tag{33}$$

where superscript (k) indicates the terms with the k-th sampling.

REFERENCES

[1] 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.

[2] X. Yuan, L. Li, Y. A. W. Shardt, Y. Wang, and C. Yang, "Deep learning with spatiotemporal attention-based LSTM for industrial soft sensor model development," *IEEE. T. Ind. Electron.*, pp. DOI: 10.1109/TIE.2020.2984443, 2020.

[3] 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.

[4] C. Brosilow, and M. Tong, "Inferential control of processes: Part II. The structure and dynamics of inferential control systems," *AIChE J.*, vol. 24, no. 3, pp. 492-500, 1978.

[5] 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.*, vol. 68, no. 5, pp. 1637 - 1646 2018.

[6] Y. Fan, B. Tao, Y. Zheng, and S. Jang, "A Data-Driven Soft Sensor Based on Multilayer Perceptron Neural Network With a Double LASSO Approach," *IEEE T. Instrum. Meas.*, vol. 69, no. 7, pp. 3972-3979, 2020.

[7] J. Corrigan, and J. Zhang, "Integrating dynamic slow feature analysis with neural networks for enhancing soft sensor performance," *Comput. Chem. Eng.*, vol. 139, pp. 106842, 2020/08/04/, 2020.

[8] 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.

[9] G. P. Morais, B. H. G. Barbosa, D. D. Ferreira, and L. S. Paiva, "Soft sensors design in a petrochemical process using an Evolutionary Algorithm," *Measurement*, vol. 148, pp. 106920, 2019.

[10] W. Shao, Z. Ge, and Z. Song, "Quality variable prediction for chemical processes based on semisupervised Dirichlet process mixture of Gaussians," *Chem. Eng. Sci.*, vol. 193, pp. 394-410, 2019.

[11] W. Shao, L. Yao, Z. Ge, and Z. Song, "Parallel Computing and SGD-Based DPMM For Soft Sensor Development With Large-Scale Semisupervised Data," *IEEE. T. Ind. Electron.*, vol. 66, no. 8, pp. 6362-6373, 2019.

[12] X. Yuan, C. Ou, Y. Wang, C. Yang, and W. Gui, "Deep quality-related feature extraction for soft sensing modeling: A deep learning approach with hybrid VW-SAE," *Neurocomputing*, vol. 396, pp. 375-382, 2020.

[13] P. Czop, G. Kost, D. Sławik, and G. Wszołek, "Formulation and identification of First- Principle Data-Driven models," *Journal of Achievements in Materials & Manufacturing Engineering*, vol. 44, no. 2, pp. 2718-2723, 2011.

[14] V. Prasad, M. Schley, L. P. Russo, and B. Wayne Bequette, "Product property and production rate control of styrene polymerization," *J. Process Contr.*, vol. 12, no. 3, pp. 353-372, 2002/04/01/, 2002.

[15] P. Kadlec, B. Gabrys, and S. Strandt, "Data-driven soft sensors in the process industry," *Comput. Chem. Eng.*, vol. 33, no. 4, pp. 795-814, 2009.

[16] R. Jia, Z. Mao, Y. Chang, and L. Zhao, "Soft-sensor for copper extraction process in cobalt hydrometallurgy based on adaptive hybrid model," *Chemical Engineering Research and Design*, vol. 89, no. 6, pp. 722-728, 2011/06/01/, 2011.

[17] J. Dai, N. Chen, X. Yuan, W. Gui, and L. Luo, "Temperature prediction for roller kiln based on hybrid first-principle model and data-driven MW-DLWKPCR model," *ISA T.*, vol. 98, pp. 403-417, 2020.

[18] N. Chen, J. Dai, X. Yuan, W. Gui, W. Ren, and H. N. Koivo, "Temperature Prediction Model for Roller Kiln by ALD-Based Double Locally Weighted Kernel Principal Component Regression " *IEEE T. Instrum. Meas.*, vol. 67, no. 8, pp. 2001 - 2010 2018.

[19] X. Yan, "Hybrid artificial neural network based on BP-PLSR and its application in development of soft sensors," *Chemometr. Intell. Lab. Syst*, vol. 103, no. 2, pp. 152-159, 2010.

[20] H. Jin, X. Chen, J. Yang, and L. Wu, "Adaptive soft sensor modeling framework based on just-in-time learning and kernel partial least squares regression for nonlinear multiphase batch processes," *Comput. Chem. Eng.*, vol. 71, pp. 77-93, 2014/12/04/, 2014.

[21] H. Jin, X. Chen, J. Yang, H. Zhang, L. Wang, and L. Wu, "Multi-model adaptive soft sensor modeling method using local learning and online support vector regression for nonlinear time-variant batch processes," *Chem. Eng. Sci.*, vol. 131, pp. 282-303, 2015.

[22] W. Shao, Z. Ge, Z. Song, and K. Wang, "Nonlinear industrial soft sensor development based on semi-supervised probabilistic mixture of extreme learning machines," *Control Eng. Pract.*, vol. 91, pp. 104098, 2019/10/01/, 2019.

[23] Z. Zhang, and J. Zhao, "A deep belief network based fault diagnosis model for complex chemical processes," *Comput. Chem. Eng.*, vol. 107, pp. 395-407, 2017.

[24] X. Yuan, J. Zhou, B. Huang, Y. Wang, C. Yang, and W. Gui, "Hierarchical quality-relevant feature representation for soft sensor modeling: a novel deep learning strategy," *IEEE. T. Ind. Inf.*, vol. 16, no. 6 pp. 3721-3730, 2020.

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

[26] L. Yao, and Z. Ge, "Deep Learning of Semisupervised Process Data With Hierarchical Extreme Learning Machine and Soft Sensor Application," *IEEE. T. Ind. Electron.*, vol. 65, no. 2, pp. 1490-1498, 2018.

[27] L. Yao, and Z. Ge, "Distributed parallel deep learning of Hierarchical Extreme Learning Machine for multimode quality prediction with big process data," *Eng. Appl. Artif. Intel*, vol. 81, pp. 450-465, 2019.

[28] Y. Liu, C. Yang, Z. Gao, and Y. Yao, "Ensemble deep kernel learning with application to quality prediction in industrial polymerization processes," *Chemometr. Intell. Lab. Syst*, vol. 174, pp. 15-21, 2018.

[29] Y. Liu, F. Yu, and J. Chen, "Flame Images for Oxygen Content Prediction of Combustion Systems Using DBN," *Energ. Fuel.*, vol. 31, no. 8, pp. 8776-8783, 2017.

[30] 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.

[31] Y. Wang, Z. Pan, X. Yuan, C. Yang, and W. Gui, "A novel deep learning based fault diagnosis approach for chemical process with extended deep belief network," *ISA T.*, vol. 96, pp. 457-467, 2020.

[32] J. Ji, Y. Sun, F. Kong, and Q. Miao, "A Construction Approach to Prediction Intervals Based on Bootstrap and Deep Belief Network," *IEEE Access*, vol. 7, pp. 124185-124195, 2019.

[33] C. Zhu, and J. Zhang, "Developing Soft Sensors for Polymer Melt Index in an Industrial Polymerization Process Using Deep Belief Networks," *International Journal of Automation and Computing*, vol. 17, no. 1, pp. 44-54, 2020.

[34] M. G. Xibilia, M. Latino, Z. Marinković, A. Atanasković, and N. Donato, "Soft Sensors based on Deep Neural Networks for Applications in Security and Safety," *IEEE T. Instrum. Meas.*, pp. 1-1, 2020.

[35] D. Chen, J. Lv, and Z. Yi, "Graph Regularized Restricted Boltzmann Machine," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 6, pp. 2651-2659, 2018.

[36] G. E. Hinton, "A Practical Guide to Training Restricted Boltzmann Machines," *Momentum*, vol. 9, no. 1, pp. 599-619, 2012.

[37] G. E. Hinton, "Training Products of Experts by Minimizing Contrastive Divergence," *Neural Computation*, vol. 14, no. 8, pp. 1771-1800, 2002.

[38] R. J. Tibshirani, and B. Efron, "An introduction to the bootstrap," *Monographs on statistics and applied probability*, vol. 57, pp. 1-436, 1993.

[39] L. Fortuna, S. Graziani, A. Rizzo, and M. G. Xibilia, *Soft sensors for monitoring and control of industrial processes*: Springer Science & Business Media, 2007.

[40] Y. Bengio, "Practical Recommendations for Gradient-Based Training of Deep Architectures," *Neural networks: Tricks of the trade*, Berlin, Heidelberg: Springer, 2012.

[41] X. Yuan, C. Ou, Y. Wang, C. Yang, and W. Gui, "A Layer-Wise Data Augmentation Strategy for Deep Learning Networks and Its Soft Sensor Application in an Industrial Hydrocracking Process," *IEEE T. Neur. Net. Lear. Syst.*, pp. 1-10, 2019.

[42] X. Yuan, J. Zhou, and Y. Wang, "A Comparative Study of Adaptive Soft Sensors for Quality Prediction in an Industrial Refining Hydrocracking Process," in 2018 IEEE 7th Data Driven Control and Learning Systems Conference (DDCLS), Enshi, China, 2018, pp. 1064-1068.