ISA Transactions xxx (xxxx) xxx

Contents lists available at ScienceDirect

ISA Transactions

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

Research article

A novel deep learning based fault diagnosis approach for chemical process with extended deep belief network

Yalin Wang, Zhuofu Pan, Xiaofeng Yuan *, Chunhua Yang, Weihua Gui

School of Automation, Central South University, Changsha, 410083, Hunan, PR China

HIGHLIGHTS

- An extended DBN is proposed for feature extraction and fault classification.
- Raw data is combined with hidden features at previous ERBM as inputs for next one.
- EDBN is beneficial for retaining enough valuable information from raw data.
- High classification performance of the extended DBN is validated on TE process.

ARTICLE INFO

Article history:
Received 25 January 2019
Received in revised form 1 July 2019
Accepted 1 July 2019
Available online xxxx

Keywords:
Fault detection and diagnosis
Deep learning
Deep belief network
Extended DBN

ABSTRACT

Deep learning networks have been recently utilized for fault detection and diagnosis (FDD) due to its effectiveness in handling industrial process data, which are often with high nonlinearities and strong correlations. However, the valuable information in the raw data may be filtered with the layer-wise feature compression in traditional deep networks. This cannot benefit for the subsequent fine-tuning phase of fault classification. To alleviate this problem, an extended deep belief network (EDBN) is proposed to fully exploit useful information in the raw data, in which raw data is combined with the hidden features as inputs to each extended restricted Boltzmann machine (ERBM) during the pre-training phase. Then, a dynamic EDBN-based fault classifier is constructed to take the dynamic characteristics of process data into consideration. Finally, to test the performance of the proposed method, it is applied to the Tennessee Eastman (TE) process for fault classification. By comparing EDBN and DBN under different network structures, the results show that EDBN has better feature extraction and fault classification performance than traditional DBN.

1. Introduction

Thanks to advanced process control systems, modern chemical processes have become highly automated, which allows continuous production with low cost and high safety. However, even for chemical plants equipped with distributed control systems and safety instrument systems, there still have many accidents reported in recent years, which can easily result in serious fatalities, asset damage and environmental destruction [1,2]. This is due to the reason that it often largely dependent on process operators to monitor and handle abnormal situations in chemical processes. However, it is difficult to ensure that operators can always discover abnormal situations and make the right operation timely. Therefore, as a real-time, reliable and efficient advanced monitoring means, fault detection and diagnosis (FDD) [3] has played an important role in process monitoring and system fault tracking, which has received extensive attention from academia and industry. Moreover, FDD can give valuable information for timely process control and optimization [4–10].

In FDD, fault detection aims to monitor a system and identify when a fault occurs in the processes. Fault diagnosis concerns about which fault type the new identified fault status belongs to and what the root fault cause is. Hence, it is an important step to carry out fault classification to identify the detailed fault category. Generally, fault classification can be treated as a multi-classification task, which can not only detect whether there is a fault, but also identify the category of the fault type. Typically, fault classification approaches can be divided into the model-based, knowledge-based and data-driven approaches [3,11]. Compared to model-based and knowledge-based methods, which strongly depend on process physicochemical knowledge, data-driven methods can build fault classification models by extracting valuable information from process history data. By far, it has become more and more popular for data-driven methods in

* Corresponding author.
E-mail addresses: ylwang@csu.edu.cn (Y. Wang), panfuzz@csu.edu.cn (Z. Pan), yuanxf@csu.edu.cn (X. Yuan), ychh@csu.edu.cn (C. Yang), gwh@csu.edu.cn (W. Gui).

https://doi.org/10.1016/j.isatra.2019.07.001
0019-0578/

handling fault classification problems due to the large amount of data that can be collected in modern chemical plants.

In general, the massive process data collected in chemical plants are with high redundancy and strong correlations, in which valuable information is submerged and needs to be mined effectively. How to extract useful information is the key step to establish a stable and robust model, as well as improve the classification performance [12]. During the past years, many feature extractors have been developed based on machine learning approaches and multivariate statistical methods. Canonical correlation analysis (CCA) [13,14], principal component analysis (PCA) [15–17], Fisher discriminant analysis (FDA) [18] and partial least squares (PLS) [19], and are some of the most widely used linear feature representation methods. Alternatively, nonlinear feature extractors like kernel PCA [20], kernel PLS [21], artificial neural networks (ANN) [22,23] and support vector machine (SVM) [24] have also been applied for describing more complicated data characteristics. These traditional feature extractors can be regarded as shallow learning networks, which learn useful features for many pattern recognition tasks. However, they are limited in their representation for massive data in large-scale complex processes, especially in the era of big data. In contrast, deep multi-layer networks are more capable of having excellent expression capability, which can extract more useful and predictive features for highly complex systems.

Actually, multi-layer networks did not function well in the early days owing to the gradient exploding and vanishing problems during the network training stage. This bottleneck is not broken until the deep learning technique was developed by Hinton et al. in 2006, with which the deep networks are first unsupervised layer-wise pre-trained and then fine-tuned by back-propagation (BP) algorithm. Since then, deep learning has raised great attention in areas like speech recognition, image recognition, and natural language processing, in which it has shown powerful ability in identifying and expressing deep features with different complex tasks [25].

In the last five years, deep learning networks have also been used in chemical industry for process data modeling like soft sensor and fault classification applications [26–30]. For example, Jiang et al. [31] utilized stacked denoising auto-encoder (SDAE) to build a chemical fault classification model, in which an active learning strategy was proposed that allows the model to select the informative data for online fine-tuning. Then, Zhang et al. [11] introduced deep belief network (DBN) for fault classification in benchmarked Tennessee Eastman (TE) process. They employed mutual information approach to extract features and established DBN sub-networks for each state in TE process. Also, Jiang et al. [32] applied stacked sparse auto-encoders (SSAE) for TE process fault classification. They properly considered the dynamic characteristics of the process data and provided a semi-supervised learning strategy for the deep network. Moreover, Wu et al. [1] exploited convolutional neural network (CNN) in TE process for fault classification application. Recently, Yu et al. [33] used DBN to monitor and enhance the abnormal fluctuation information, and the enhanced features extracted via DBN are used for fault detection in TE process. These methods show the great potential and outstanding performance in process data modeling with deep learning technique.

Although many research works have been published on deep learning for fault classification problems, there are still some specific limitations in existing deep learning models. For instance, the reported deep-learning-based techniques are often directly used to divide fault category without seeking a better feature representation to fit the dynamic and nonlinear characteristics of industrial data. In addition, as hierarchical features are extracted layer by layer in deep learning, successive feature compression may result in the loss of valuable information in the raw data, which is disadvantageous for fault classification in the subsequent fine-tuning phase. To fully exploit the potential valuable information in the raw data, an extended deep belief network (EDBN) is designed for feature representation and fault classification in this paper. In EDBN, features are progressively extracted through stacking multiple extended restricted Boltzmann machines (ERBM) during the pre-training phase. Then, the fault category results are outputted by an additional classification layer located at the top hidden layer of the last ERBM in the fine-tuning phase. For each extended RBM (ERBM), the hidden features from the previous RBM/ERBM are combined with the raw input data to serve as the new inputs to the current ERBM. By adding the raw data to each ERBM, the layer-wise features related to the raw data can be progressively extracted, which is helpful to mine potential valuable information in the raw data. Finally, to take the important process dynamics into consideration, a dynamic EDBN-based modeling framework is further built for fault classification. The performance of the designed EDBN is shown on the well-known benchmarked TE process.

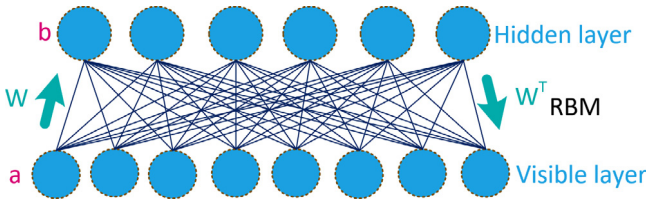

Fig. 1. The diagram of network structure for RBM.

The rest of the paper is structured as follows. Section 2 simply revisits the structure of RBM and DBN. The extended deep belief network is then introduced and explained in Section 3, in which the detailed procedure of EDBN-based fault classification modeling is also described. After that, the proposed EDBN is utilized for fault classification on the TE benchmark in Section 4. At last, conclusions are summarized for this paper in Section 5.

2. Deep belief network

2.1. Restricted Boltzmann machine

Restricted Boltzmann machine (RBM) [34], as the basic module of deep belief network, mainly consists of one visible layer and a hidden layer. Different from Boltzmann machine (BM) with a fully connected graph structure, RBM limits the interconnection of peer nodes to ensure their mutual independence. The network structure of RBM is given in Fig. 1.

RBM is a probabilistic model, whose parameters are composed of the weights and biases. Assume the visible and hidden variable vectors of the RBM are $\mathbf{v}$ and $\mathbf{h}$. Their joint probability density can be represented as

$$p(\mathbf{v},\mathbf{h}) = e^{-E(\mathbf{v},\mathbf{h})} \Big/ \iint_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}, \quad (1)$$

where $E(\mathbf{v},\mathbf{h})$ is the so-called energy function, whose form depends on the unit type of RBM. The commonly used types are binary unit and Gaussian unit. When dealing with continuous-value data, a good choice is to apply Gaussian–Gaussian energy function (Eq. (2)) rather than binary–binary energy function (Eq. (3)). The forms can be expressed as [35]

$$E(\mathbf{v},\mathbf{h}) = \sum_{i\in vis} \frac{(v_i - a_i)^2}{2\sigma_i^2} + \sum_{j\in hid} \frac{(h_j - b_j)^2}{2\sigma_j^2} - \sum_{i\in vis, j\in hid} \frac{v_i}{\sigma_i}\frac{h_j}{\sigma_j} w_{ij}, \quad (2)$$

Please cite this article as: Y. Wang, Z. Pan, X. Yuan et al., A novel deep learning based fault diagnosis approach for chemical process with extended deep belief network. ISA Transactions (2019), https://doi.org/10.1016/j.isatra.2019.07.001.

$$E(\mathbf{v},\mathbf{h}) = -\sum_{i\in vis} a_i v_i - \sum_{j\in hid} b_j h_j - \sum_{i\in vis, j\in hid} v_i h_j w_{ij}, \quad (3)$$

where h_j and v_i represent the activation states of hidden layer unit j and visible layer unit i, respectively; b_j and a_i refer to their corresponding bias terms; w_{ij} is the weight that connects v_i and h_j. In Gaussian energy function, σ_i and σ_j are the standard deviation terms of the Gaussian noises introduced in the ith visible unit and the jth hidden unit, which usually take the value as $\sigma_i = \sigma_j = 1$. With the joint probabilistic distribution $p(\mathbf{v},\mathbf{h})$ and its marginal probabilistic distributions $p(\mathbf{v})$, $p(\mathbf{h})$, the conditional probabilistic distribution $p(\mathbf{h}|\mathbf{v})$, $p(\mathbf{v}|\mathbf{h})$ can be obtained by Bayesian inference as

$$\begin{aligned} p(\mathbf{h}|\mathbf{v}) &= p(\mathbf{v},\mathbf{h})/p(\mathbf{v}) = e^{-E(\mathbf{v},\mathbf{h})} \Big/ \int_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \\ p(\mathbf{v}|\mathbf{h}) &= p(\mathbf{v},\mathbf{h})/p(\mathbf{h}) = e^{-E(\mathbf{v},\mathbf{h})} \Big/ \int_{\mathbf{v}} e^{-E(\mathbf{v},\mathbf{h})} \end{aligned}. \quad (4)$$

For the Gaussian units, their conditional distribution should obey a normal distribution with following forms [11]

$$\begin{aligned} p(h_j|\mathbf{v}) &\sim N(\mu_j,\sigma_j),\ \mu_j = b_j + \sigma_j \sum_{i\in vis} \frac{v_i}{\sigma_i} w_{ij} = line(\mathbf{v}) \\ p(v_i|\mathbf{h}) &\sim N(\mu_i,\sigma_i),\ \mu_i = a_i + \sigma_i \sum_{j\in hid} \frac{h_j}{\sigma_j} w_{ij} = line(\mathbf{h}) \end{aligned}. \quad (5)$$

2.2. Deep belief network

Deep belief network is composed of multiple stacked RBMs and an output layer added on the last RBM, the structure of which is illustrated in Fig. 2. The training procedure of DBN includes two stages: the layer-wise unpervised pre-training and fine-tuning. During the pre-training phase, a layer-wise greedy technique is employed to train the RBMs. Once the training of the previous RBM is completed, its hidden layer is used as the visible layer for the next RBM. In such a way, RBMs can be trained one by one until the last RBM is trained. Each RBM is trained by maximizing the probability of its input data, in which contrastive divergence (CD) algorithm [36,37] is exploited to update the parameters. After pretraining, a classification layer is added to the last hidden layer and DBN will be further fine-tuned via minimizing the error between estimated output values and labels. The BP algorithm is implemented to progressively pass the error from the last layer to the bottom input layer. In this way, the parameters of the whole network can be updated.

3. Extended deep belief network for fault classification

3.1. Extended deep belief network

Although DBN can extract effective deep features and achieve fast convergence by performing pre-training and fine-tuning, there is still room for improvement of learning performance. According to the information bottleneck theory, as the number of neural network layers increases, the relevant information between the extracted deep features and the raw data will be less and less. Therefore, in the layer-wise compression procedure of most existing deep networks, a lot of useful information in the raw data may be usually lost in high layers. To alleviate this problem, an extended deep belief network (EDBN) is proposed to sufficiently capture the valuable information in the raw data, which is stacked by multiple ERBMs. By utilizing the raw data as additional inputs to the visible layer of each ERBM for pre-training, the raw input data can participate in the whole compression procedure. Thus, the extracted deep features are highly related with the raw data, where the potential valuable information is fully reserved. Compared with existing methods, EDBN can repeatedly distill valuable information from raw data and can provide deep compressed representations that are highly correlated with the raw data. Moreover, EDBN can achieve higher accuracy and lower false positive rate for classification task. The structure of EDBN can be seen in Fig. 3.

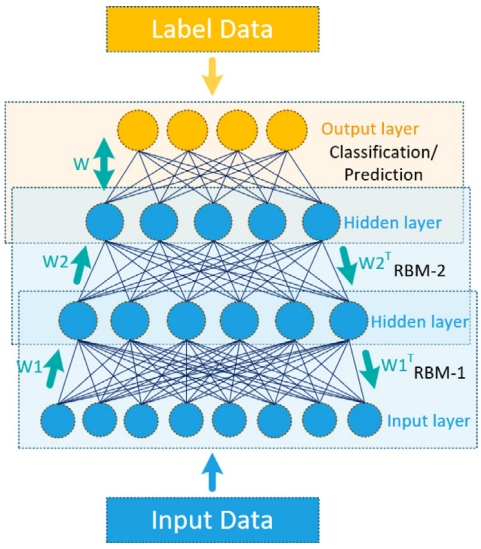

Fig. 2. An illustration of DBN with two hidden layers.

The training procedure of EDBN is also composed of pre-training and fine-tuning stages. During the pre-training stage, the raw data is added to the visible layer of each extended RBM (ERBM). In the left subfigure of Fig. 3, the inputs of each ERBM is composed of the raw data expressed by purple circles and the previous hidden features described by blue circles. Also, its weight matrix consists of two parts of $\mathbf{w}_I$ and $\mathbf{w}_H$, in which the former connects the raw data with the hidden layer, and the latter connects the hidden feature of the previous ERBM with the hidden layer. Then, the maximum likelihood rule and CD algorithm will be used to update the parameters of each ERBM. In this way, network parameters can be well learned, and their values will be used as the initialization for the fine-tuning. In the fine-tuning stage, another output layer is added for classification, and the extended raw variable nodes will be dropped out in each hidden layer. Finally, the loss between predicted outputs and labels will be calculated and BP algorithm is iteratively executed to update the network parameters by minimizing the loss function. The detailed training procedures are described in Sections 3.2 and 3.3.

3.2. Pre-training

The pre-training procedure of EDBN is to train these ERBMs one by one. As for the first ERBM, it is actually an RBM since there is no need to extend raw data. Every ERBM updates their weight and biases based on the maximum likelihood theory and k-step contrastive divergence learning (CD-k) algorithm [38,39]. Generally, maximum likelihood method can be applied to obtain proper network parameters ($\boldsymbol{\theta} = \{\mathbf{w},\mathbf{a},\mathbf{b}\}$ in ERBM) that best fit the distribution of the input data. By calculating the logarithmic partial derivatives of $p(\mathbf{v} = \mathbf{v}_{data})$, the gradient update formula of network parameters can be obtained as Eq. (6), which is given by Fischer [40]. Since it is quite difficult to calculate the second terms in Eq. (6) accurately, CD-k algorithm based on Markov Chain and Monte Carlo sampling is employed to obtain

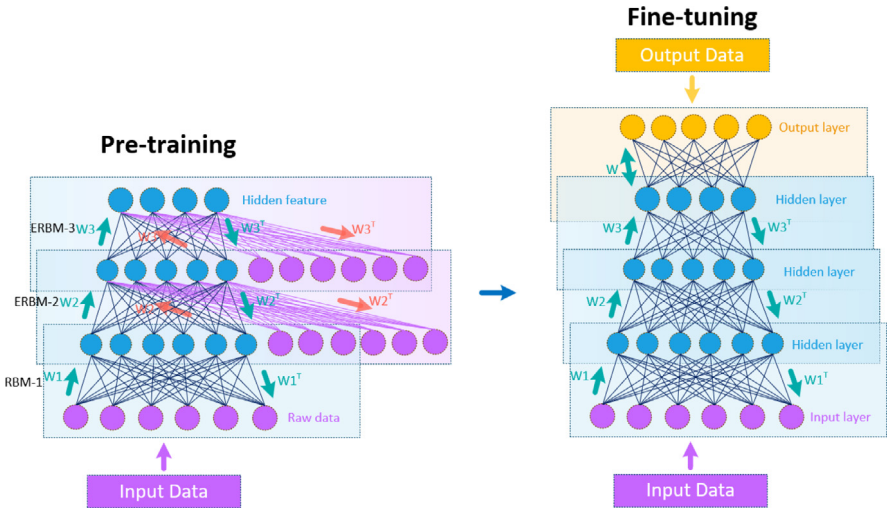

Fig. 3. The structure of EDBN during the pre-training phase and fine-tuning phase. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the approximate solution of this problem. The main idea is to transfer the data between visible and hidden layer for k times, so that the network states $\mathbf{h}$ and $\mathbf{v}$ can mostly represent the distribution of the model after k iterations. As shown in Fig. 4, the input of the ERBM $\mathbf{v}_{data}$ can be expressed as $\mathbf{v}^{(0)}$. Via sampling from the probability $p\left(\mathbf{h}|\mathbf{v}^{(0)}\right)$, the state of the hidden units can be obtained as $\mathbf{h}^{(0)}$. Similarly, $\mathbf{v}^{(1)}$ can be obtained via sampling from $p\left(\mathbf{v}|\mathbf{h}^{(0)}\right)$. The learning process from $\mathbf{v}^{(0)}$ to $\mathbf{v}^{(1)}$ is also called one step Gibbs sampling. According to Markov theory, after k-step Gibbs sampling ($k \to \infty$), it will converge to the stationary distribution, and $\mathbf{v}^{(k)}$ at this time will reflect the distribution of model. By further approximating the expectations over $p(\mathbf{v})$ in Eq. (6) with the single sample $\mathbf{v}^{(k)}$ (Monte Carlo sampling), Eq. (6) can be simply converted to Eq. (7) as

$$\frac{\partial \log p\left(\mathbf{v}=\mathbf{v}_{data}\right)}{\partial \boldsymbol{\theta}} = -\int_{\mathbf{h}} p\left(\mathbf{h}|\mathbf{v}_{data}\right)\frac{\partial E\left(\mathbf{v}_{data},\mathbf{h}\right)}{\partial \boldsymbol{\theta}} + \int_{\mathbf{v}} p\left(\mathbf{v}\right)\int_{\mathbf{h}} p\left(\mathbf{h}|\mathbf{v}\right)\frac{\partial E\left(\mathbf{v},\mathbf{h}\right)}{\partial \boldsymbol{\theta}}, \tag{6}$$

$$CD_k\left(\boldsymbol{\theta},\mathbf{v}^{(0)}\right) = -\frac{\partial E\left(\mathbf{v}^{(0)},\mathbf{h}^{(0)}\right)}{\partial \boldsymbol{\theta}} + \frac{\partial E\left(\mathbf{v}^{(k)},\mathbf{h}^{(k)}\right)}{\partial \boldsymbol{\theta}}. \tag{7}$$

In Eq. (7), the first and second terms are called negative term and positive term, respectively. They can reflect the raw data distribution and the model distribution. In the training process, the CD step can simply take $k = 1$, which can meet the requirements of calculation accuracy. As mentioned in Section 2.1, if the units type of the RBM/ERBM are Gaussian-Gaussian, then their conditional probabilistic distribution $p(\mathbf{h}|\mathbf{v})$ and $p(\mathbf{v}|\mathbf{h})$ will obey a normal density. Thus, the model states $\mathbf{v}$ and $\mathbf{h}$ can take the means of the normal distributions as their sampling values.

By substituting Eq. (2) into Eq. (7), Eqs. (8)–(10) can be further obtained. Then, the parameters $\boldsymbol{\theta}_{pre} = \{\mathbf{w}_{pre}, \mathbf{a}_{pre}, \mathbf{b}_{pre}\}$ of ERBM can be updated by Eqs. (8) to (11) as

$$\Delta w_{ij} = v_i^{(0)}h_j^{(0)} - v_i^{(k)}h_j^{(k)}, \tag{8}$$

$$\Delta a_i = v_i^{(0)} - v_i^{(k)}, \tag{9}$$

$$\Delta b_j = h_j^{(0)} - h_j^{(k)}, \tag{10}$$

$$\boldsymbol{\theta}_{pre}^{(epoch+1)} = \boldsymbol{\theta}_{pre}^{(epoch)} + m\Delta\boldsymbol{\theta}_{pre}^{(epoch-1)} + r\Delta\boldsymbol{\theta}_{pre}^{(epoch)}, \tag{11}$$

where m is the momentum for accelerating the learning procedure; r is the learning rate; *epoch* represents the epoch of iterations; weight matrix consists of two parts as $\mathbf{w}_{pre} = [\mathbf{w}_H, \mathbf{w}_l]$. In algorithm 1, a batch version of CD-k is explained.

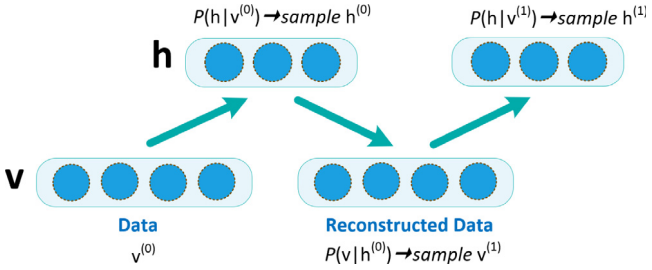

Fig. 4. The process of training RBM/ERBM with contrast divergence algorithm.

The pre-training process for the whole EDBN is described as follows. Assume $\mathbf{v}^{(i,j)}$ and $\mathbf{h}^{(i,j)}$ stand for the activation states of the visible and hidden units in the i th ERBM at the j th Gibbs sampling step, respectively. As shown in Fig. 5, the raw observed data $\mathbf{v}^{(1,0)}$ is first fed into the input layer of RBM-1. Then, hidden features $\mathbf{h}^{(1,0)}$ are generated by sampling from conditional probability $p\left(\mathbf{h}|\mathbf{v}^{(1,0)}\right)$. After that, $\mathbf{v}^{(1,1)}$ and $\mathbf{h}^{(1,1)}$ can be available by further sampling from $p\left(\mathbf{v}|\mathbf{h}^{(1,0)}\right)$ and $p\left(\mathbf{h}|\mathbf{v}^{(1,1)}\right)$. By substituting the states into Eqs. (8)–(10), the parameters of the first RBM can be well trained. After RBM-1 is trained, its network parameters will be saved and used for transmitting the raw data $\mathbf{v}_{data}^{(1)}$ into hidden feature $\mathbf{h}_{data}^{(1)}$. Then, the raw data and the hidden feature of RBM-1 are combined as the inputs $\mathbf{v}_{data}^{(2)} = [\mathbf{h}_{data}^{(1)}, \mathbf{v}_{data}^{(1)}]$ for ERBM-2, and ERBM-2 will be trained in the same manner. In such a way, the remaining ERBMs will be trained one by one until the last one is trained. Finally, the features of the raw data will be extracted in a hierarchical and progressive manner, and the well-trained parameters will be used as the initial value for further fine-tuning. The whole pre-training process of EDBN can be summarized in algorithm 2.

3.3. Fine-tuning

During the fine-tuning phase of EDBN, an additional layer for output is added at the last hidden layer of EDBN to obtain the probabilities of the sample into different categories. The parameters of hidden layers in EDBN $\{\mathbf{w}_{ft}^{(i)}, \mathbf{b}_{ft}^{(i)}\}_{i=1,2,\ldots,l}$ are initialized by

Please cite this article as: Y. Wang, Z. Pan, X. Yuan et al., A novel deep learning based fault diagnosis approach for chemical process with extended deep belief network. ISA Transactions (2019), https://doi.org/10.1016/j.isatra.2019.07.001.

Algorithm 1: k-step contrastive divergence (CD-k)

Input: $RBM(\boldsymbol{v}, \boldsymbol{h})$, training batch $\boldsymbol{B}$

Output: approximate gradient $\Delta \boldsymbol{w}, \Delta \boldsymbol{a}, \Delta \boldsymbol{b}$

init $\Delta w_{ij} = \Delta a_i = \Delta b_j = 0$ *for* $i = 1, ..., n, j = 1, ..., m$

forall the *sample* $\in \boldsymbol{B}$ **do**

  $\boldsymbol{v}^{(0)} \leftarrow$ ***sample***

  for $t = 0, ..., k-1$ **do**

    for $j = 1, ..., m$ **do** sample $h_j^{(t)}$ from $p(h_j \mid \boldsymbol{v}^{(t)})$

    for $i = 1, ..., n$ **do** sample $v_i^{(t+1)}$ from $p(v_i \mid \boldsymbol{h}^{(t)})$

  sample $h_j^{(k)}$ from $p(h_j \mid \boldsymbol{v}^{(k)})$ *for* $j = 1, ..., m$

  for $i = 1, ..., n, j = 1, ..., m$ **do**

    $\Delta w_{ij} \leftarrow \Delta w_{ij} + v_i^{(0)} \cdot h_j^{(0)} - v_i^{(k)} \cdot h_j^{(k)}$

    $\Delta a_i \leftarrow \Delta a_i + v_i^{(0)} - v_i^{(k)}$

    $\Delta b_j \leftarrow \Delta b_j + h_j^{(0)} - h_j^{(k)}$

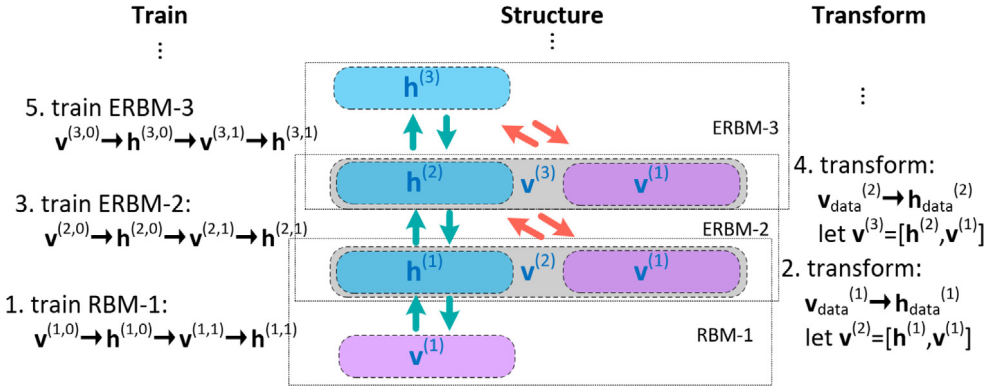

Fig. 5. Pre-training of EDBN: training RBM/ERBM layer by layer.

the pre-trained parameters $\{\mathbf{w}_H^{(i)}, \mathbf{b}_{pre}^{(i)}\}_{i=1,2,...,l}$ as

$$\begin{cases} \mathbf{w}_{ft}^{(i)} = \mathbf{w}_H^{(i)} \\ \mathbf{b}_{ft}^{(i)} = \mathbf{b}_{pre}^{(i)} \end{cases}, \quad i = 1, 2, \ldots, l, \tag{12}$$

where l represents the number of hidden layers; the extended raw data nodes and their corresponding parameters $\{\mathbf{w}_l^i\}_{i=2,3,...,l}$ are dropped out after pretraining. For the parameters $\{\mathbf{w}_{ft}^{(o)}, b_{ft}^{(o)}\}$ of the output layer, random values are initialized for them. Therefore, the parameters need to be fine-tuned are $\boldsymbol{\theta}_{ft} = \{\mathbf{w}_{ft}^{(i)}, \mathbf{b}_{ft}^{(i)}, \mathbf{w}_{ft}^{(o)}, \mathbf{b}_{ft}^{(o)}\}_{i=1,2,...,l}$. Through forward propagation, the classification loss error $\mathbf{C}$ is calculated between the predicted outputs and the real labels. Then, the partial derivatives of the loss function with regard to the network parameters $\Delta\boldsymbol{\theta}_{ft} = \partial\mathbf{C}/\partial\boldsymbol{\theta}_{ft}$ can be further obtained by the BP algorithm. At last, based on the adaptive moment estimation (Adam) algorithm proposed by Kingma [41], the parameters of EDBN can be further tuned with Eq. (13) to (16).

$$r^{(epoch+1)} = r^{(0)} \sqrt{\left(1 - \beta_2^{epoch}\right) / \left(1 - \beta_1^{epoch}\right)}, \tag{13}$$

$$\mathbf{m}_1^{(epoch+1)} = \beta_1 \mathbf{m}_1^{(epoch)} + (1 - \beta_1)\, \Delta\boldsymbol{\theta}_{ft}^{(epoch)}, \tag{14}$$

$$\mathbf{m}_2^{(epoch+1)} = \beta_2 \mathbf{m}_2^{(epoch)} + (1 - \beta_2) \left(\Delta\boldsymbol{\theta}_{ft}^{(epoch)}\right)^2, \tag{15}$$

$$\boldsymbol{\theta}_{ft}^{(epoch+1)} = \boldsymbol{\theta}_{ft}^{(epoch)} - r^{(epoch+1)} \mathbf{m}_1^{(epoch+1)} / \sqrt{\mathbf{m}_2^{(epoch+1)} + \varepsilon}, \tag{16}$$

where *epoch* represents the number of iterations; $r^{(epoch)}$ refers to the learning rate, which has an initial learning rate of $r^{(0)}$ set by user; $\mathbf{m}_1^{(epoch)}$, $\mathbf{m}_2^{(epoch)}$ are first moment estimate and second raw moment estimate, respectively, both of which have an initial values of $\mathbf{0}$; $\beta_1, \beta_2, \varepsilon$ are the parameters of Adam, which are recommended to set to 0.9, 0.999, 10^{-8}, separately.

3.4. Dynamic EDBN-based fault classification model

The proposed EDBN model can be used to extract deep features and predict classification results in the fault classification task of industrial chemical processes. As process data often have strong temporal correlations, the established fault classification model must consider the dynamic characteristics of process data.

Please cite this article as: Y. Wang, Z. Pan, X. Yuan et al., A novel deep learning based fault diagnosis approach for chemical process with extended deep belief network. ISA Transactions (2019), https://doi.org/10.1016/j.isatra.2019.07.001.

Algorithm 2: pre-training *EDBN* layer-by-layer

Input: *EDBN*, training set $\boldsymbol{X}$

Output: pre-trained *EDBN*

forall the *RBM/ERBM* **in** *EDBN* **do**
  init network parameters $\boldsymbol{w}, \boldsymbol{a}, \boldsymbol{b}$
  if the model to be trained **is** *RBM* **then** $\boldsymbol{Input} \leftarrow \boldsymbol{X}$
  else $\boldsymbol{Input} \leftarrow$ combine $\boldsymbol{H}$ with $\boldsymbol{X}$
  for $epoch = 1, ..., e$ **do**
    for $k = 1,...,floor(N_{sample}/N_{batch\,size})$ **do**
      $\boldsymbol{B} \leftarrow$ take a batch from $\boldsymbol{Input}$
      $\Delta\boldsymbol{w}, \Delta\boldsymbol{a}, \Delta\boldsymbol{b} \leftarrow$ **algorithm 1:** CD-k
      $\boldsymbol{w} \leftarrow \boldsymbol{w} + r \cdot \Delta\boldsymbol{w}$
      $\boldsymbol{a} \leftarrow \boldsymbol{a} + r \cdot \Delta\boldsymbol{a}$
      $\boldsymbol{b} \leftarrow \boldsymbol{b} + r \cdot \Delta\boldsymbol{b}$
  $\boldsymbol{H} \leftarrow \boldsymbol{Input} \times \boldsymbol{w} + \boldsymbol{b}$

Table 1
Selected variables and process faults in the TE benchmark.

No.	Variable	Variable type	No.	Variable	Status
1∼22	XMEAS (1∼22)	Process measurements	0		Normal
23∼33	XMV (1∼11)	Manipulated variables	1∼21	IDV (1∼21)	Fault (1∼21)

Hence, augmented dynamic data are constructed to build a dynamic EDBN model. Fig. 6 shows the whole augmentation procedure, which can be described as follows. Assume there are m raw measured variables. Then, a single data at sampling time t can be denoted as $\mathbf{x}^{(t)} = \left[x_1^{(t)}, x_2^{(t)}, \ldots, x_m^{(t)}\right]$, which acts as the fundamental element in a dynamic data. If the time length of the dynamic augmentation is taken as n, then a dynamic data from instant $t-n+1$ to t is denoted as $\mathbf{d}^{(t)} = \left[\mathbf{x}^{(t)}, \mathbf{x}^{(t-1)}, \ldots, \mathbf{x}^{(t-n+1)}\right]$. By expanding the dynamic data matrix into a one-dimensional form, we can obtain the single input of the EDBN. In batch training, the batch data set $\mathbf{x}_{in}$ is used as the input of the EDBN in a training step, which is randomly selected from dynamic data set with a size of b. Correspondingly, the one-hot coded data set $\mathbf{y}_{out}$ with the same size of b is utilized as the labels for dynamic EDBN model.

Based on the EDBN-based classifier, the fault classification framework can be established, which includes the offline training and online classification. In offline training, dynamic data are first generated from the historical database and transferred to the dynamic form for pre-training and fine-tuning of dynamic EDBN model. After the classifier is well-trained, it will be used for online fault classification. The classifier outputs the probabilities of all fault states and takes the maximum one as the fault category. Fig. 7 shows the modeling framework of EDBN-based classifier.

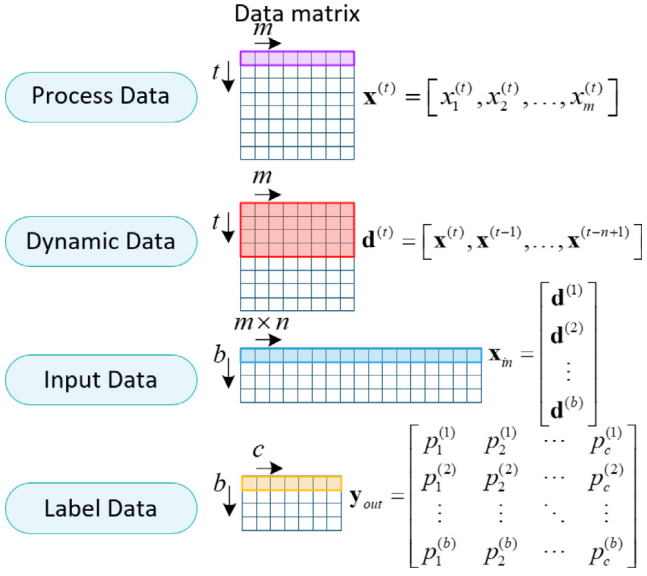

Fig. 6. The data augmentation for dynamic EDBN modeling.

4. Case study in Tennessee Eastman benchmark

4.1. Tennessee Eastman benchmark

TE process [43] is a simulated one that has been extensively used as a chemical benchmark for soft sensing, fault detection

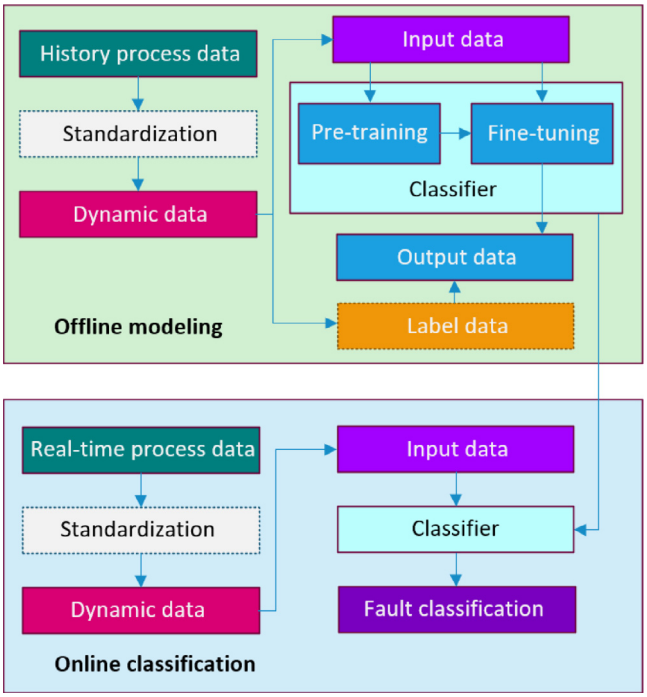

Fig. 7. The procedure of offline modeling and online classification for dynamic EDBN-based fault classification model.

and process control researches. It obtains two main products from four reactants with five major operation units: the reactor, condenser, compressor, separator and stripper. A basic illustration is provided in Fig. 8 for the Tennessee Eastman process. In TE process, there are fifty-two measured variables in total, which include nineteen composition measurements, twenty-two process measurements and eleven manipulated variables. For fault classification modeling, the first 33 measurement variables (see Table 1) are usually selected to build the classifier. The benchmark data sets can be downloaded from https://github.com/camaramm/tennessee-eastman-profBraatz. There are one normal state and 21 fault states simulated in the datasets, in which the sampling frequency is 3 min per sample. For each state, both training and testing datasets are collected. 500 training samples are recorded for the normal state and 480 training samples are recorded for each fault state. In the test data set, a total of 960 samples are collected for every state, where the fault happens at the 160th sampling instant in each fault data set. Table 2 shows the sample number in the collected data set and their dynamic augmentation with time length n.

4.2. Fault classification result and performance comparison

After the classifier model is established and well-trained, the testing dataset is utilized to assess the model performance for fault classification. Usually, the model classification performance is evaluated by the false positive rate (FPR) and fault diagnosis rate (FDR) in the normal state and each fault type [1]. FDR_i represents the correct classification rate of samples with the label of i, while FPR_i represents the misclassification proportion of samples in the other class. They are calculated as

$$FDR_i = \frac{d_i}{d_i + r_i}, \tag{17}$$

$$FPR_i = \frac{p_i}{p_i + q_i}, \tag{18}$$

where the detail symbol representation is explained in Table 3. Moreover, $\overline{FDR}$, the average fault diagnosis rate, and $\overline{FPR}$, the average false positive rate, are used for evaluating the general classification performance on the entire data set, which are defined as

$$\overline{FDR} = \frac{\sum_i d_i}{\sum_i d_i + r_i}, \tag{19}$$

$$\overline{FPR} = \frac{\sum_i p_i}{\sum_i p_i + q_i} = 1 - \overline{FDR}. \tag{20}$$

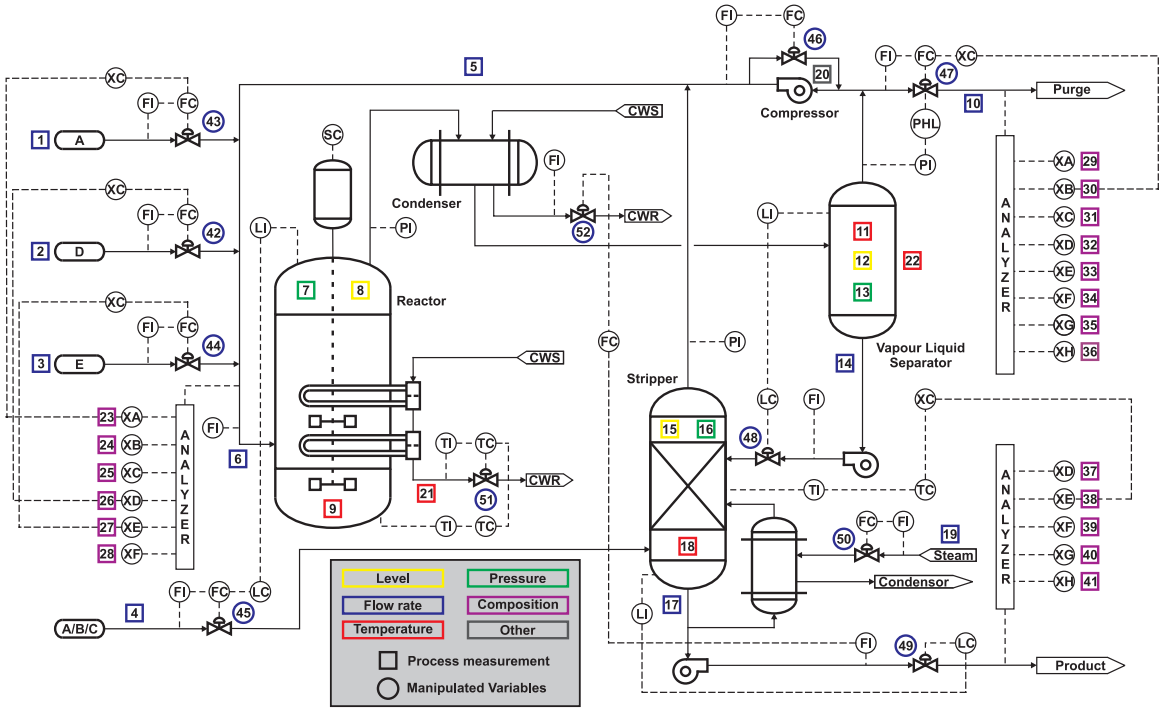

Fig. 8. The flowchart of TE process [42].

Table 2
Sample counts of TE process data set.

Status	Collected data set		Dynamic data set	
	Training	Test	Training	Test
Normal	500	960	$501-n$	$961-n$
Fault (1∼21)	480	$160_{Normal}+800_{Fault}$	$481-n$	$(161-n)_{Normal}+(801-n)_{Fault}$

Table 3
Statistical indicators for the i th class.

	Number of samples with predicted label i	Number of samples with predicted label other than i
Number of samples with real label i	d_i	r_i
Number of samples with real label other than i	p_i	q_i

Table 4
DBN/EDBN fault classification model with several candidate structures.

DBN	EDBN	Architecture	Activation function
DBN-1	EDBN-1	1320, 600, 19	**G, G**
DBN-2	EDBN-2	1320, 600, 200, 19	**G, L, G**
DBN-3	EDBN-3	1320, 600, 200, 19	**G, G, G**
DBN-4	EDBN-4	1320, 600, 400, 200, 19	**G, L, L, G**
DBN-5	EDBN-5	1320, 600, 400, 200, 19	**G, L, G, G**
DBN-6	EDBN-6	1320, 600, 400, 200, 19	**G, G, G, G**
DBN-7	EDBN-7	1320, 600, 400, 200, 100, 19	**G, L, G, L, G**
DBN-8	EDBN-8	1320, 600, 400, 200, 100, 19	**G, G, G, G, G**

The simulations were implemented on a computer with configuration as follows: 64 bit Microsoft Windows 7 operating system, Intel Xeon E5-2630 2.4 GHz processor, 32 GB RAM, and Nvidia GeForce GTX 1080 Ti Graphics card. The algorithm of DBN is developed on Anaconda platform based on the open source python package tensorflow-gpu developed by Google, which can be found from https://github.com/fuzimaoxinan/Tensorflow-Deep-Neural-Networks. The proposed EDBN algorithm is implemented based on the open DBN codes.

During the training, the epoch e is set to 35 and 240 for pre-training and fine-tuning, respectively. In each epoch, the mini-batch gradient descend strategy is utilized. Every dynamic data set $\mathbf{x}_{in}$ with a batch sample size b is fed to the model at one time. In this study, the batch size is determined as 16. Moreover, the learning rate of pre-training and fine-tuning are both 0.0001 by trial and error. In the fine-tuning phase, mean-square error (MSE) is used to calculate the loss between the estimated outputs and the label outputs. The time window length n is set to 40 to include sufficient process dynamic information. In addition, dropout strategy is for each hidden layer during the training phase, which has a dropout rate of 0.382. Furthermore, linear and Gaussian functions are selected for activation ones as

$$\begin{cases} Gaussian(x) = 1 - e^{-x^2} \\ Linear(x) = x \end{cases} \tag{21}$$

As a matter of fact, it is usually difficult to determine the best network architecture due to the lack of scientific guidance. In order to find a suitable model, some different DBN/EDBN models are taken for example for performance evaluation, the architectures of which are shown in Table 4. These network structures are designed entirely by experience, from which the most outstanding one will be selected. Parameters of the selected architecture mainly contain the number of layers and the activation function of each layer. In Table 4, "G" stands for Gaussian activation function, and "L" refers to the linear activation function whose output value is equal to the input value. Take the second network in Table 4 for example, its network structure is designed as 1330-600-200-19, which means that there are 1330 and 19 neurons in the input layer and output layer, respectively. Meanwhile, 600 and 200 refers to the number of neurons in the 1st and 2nd hidden layers, respectively. As shown in the column of activation function, "G, L" represent the first and second hidden layer of it are selected as "Gaussian" and "Linear", while the last "G" stands for the "Gaussian" activation function chosen for its output layer.

Then, the classification performance and running time spent on pre-training are compared on these networks, the results of which are listed in Table 5. In general, the performance of EDBN can outperform DBN under the same hidden network structure. DBN achieves the highest average PDR value in DBN-4 with network structure of [1320, 600, 400, 200, 19], while EDBN-2 performs best in all designed EDBN models. In terms of pre-training time, EDBN is approximately 1.02 to 1.30 times of DBN. This is because the extended raw input data to each ERBM increases the network structure and the parameters, which naturally costs more pre-training time.

From Table 5, we can find that when using the Gaussian activation function, EDBN can achieve better results even with a simple three-layer neural network. Nevertheless, with the increase of the number of hidden layers, EDBNs with more Gaussian activation function show lower average FDR than those with few Gaussian. This may be due to the reason that excessive use of nonlinear structures leads to overfitting while the original problem may not be that complicated. Hence, when using a partial of linear activation function instead of Gaussian activation function, the EDBN model achieves the best results with structure of EDBN-2.

Then, the second network structure is taken for example for further performance analysis. The detailed FDR and FPR of EDBN-2 and DBN-2 in all 19 categories are provided in Table 6. From Table 6, the detection rate of normal state and most of the fault diagnosis rates of EDBN-2 are higher than DBN-2, and the average FDR of EDBN-2 is increased by 0.42% compared with DBN-2. From the comparison of FPR, EDBN-2 can also achieve better performance since it has a lower FPR than DBN-2 almost in all categories. At last, Fig. 9 shows the detailed classification results on the test data set by taking EDBN-2 for example, where the red circles represent real labels and blue circles represent predicted ones. It can be seen from Fig. 9 that almost all the samples can be classified correctly by EDBN-2. In detail, Fig. 10 shows the proportion of samples in test data set divided into each category, where the column refers to the real labels of the data, and the row denotes the predicted one. From the simulation results, it can be concluded that EDBN-2 has superior performance in feature learning and fault classification than DBN-2.

Please cite this article as: Y. Wang, Z. Pan, X. Yuan et al., A novel deep learning based fault diagnosis approach for chemical process with extended deep belief network. ISA Transactions (2019), https://doi.org/10.1016/j.isatra.2019.07.001.

Table 5
Average FDR and time spent in the pre-training phase for several DBN/EDBN models.

DBN	Running time in pre-training (s)	$\overline{FDR}$(%)	EDBN	Running time in pre-training (s)	$\overline{FDR}$(%)
DBN-1	130	92.05	EDBN-1	133	94.18
DBN-2	236	93.89	EDBN-2	263	**94.31**
DBN-3	220	92.66	EDBN-3	252	92.88
DBN-4	307	**93.95**	EDBN-4	376	94.02
DBN-5	315	93.26	EDBN-5	379	94.22
DBN-6	311	92.24	EDBN-6	369	92.56
DBN-7	387	92.96	EDBN-7	506	94.11
DBN-8	394	92.02	EDBN-8	493	92.47

Table 6
The comparison of classification performance on DBN-2 and EDBN-2 (19-category).

Fault type	FDR (%)		FPR (%)	
	DBN-2	EDBN-2	DBN-2	EDBN-2
0 (Normal data)	87.54	**90.80**	**2.76**	3.34
1	**100**	**100**	**0**	**0**
2	**100**	**100**	**0.07**	0.08
4	**100**	**100**	**0.01**	**0.01**
5	**100**	**100**	**0**	**0**
6	**100**	**100**	**0**	**0**
7	**100**	**100**	**0**	**0**
8	**98.42**	98.29	0.3	**0.22**
10	78.19	**80.81**	0.78	**0.67**
11	**99.74**	**99.74**	**0**	**0**
12	**100**	**100**	0.03	**0**
13	90.28	**91.98**	**0**	**0**
14	**100**	**100**	0	**0**
16	**79.89**	75.56	0.97	**0.67**
17	**100**	**100**	**0**	**0**
18	**93.56**	93.43	**0**	**0**
19	**97.37**	95.53	0.56	**0.27**
20	93.04	**93.17**	0.02	**0**
21	**85.28**	83.44	1.3	**1.17**
Average	93.89	**94.31**	6.11	**5.69**

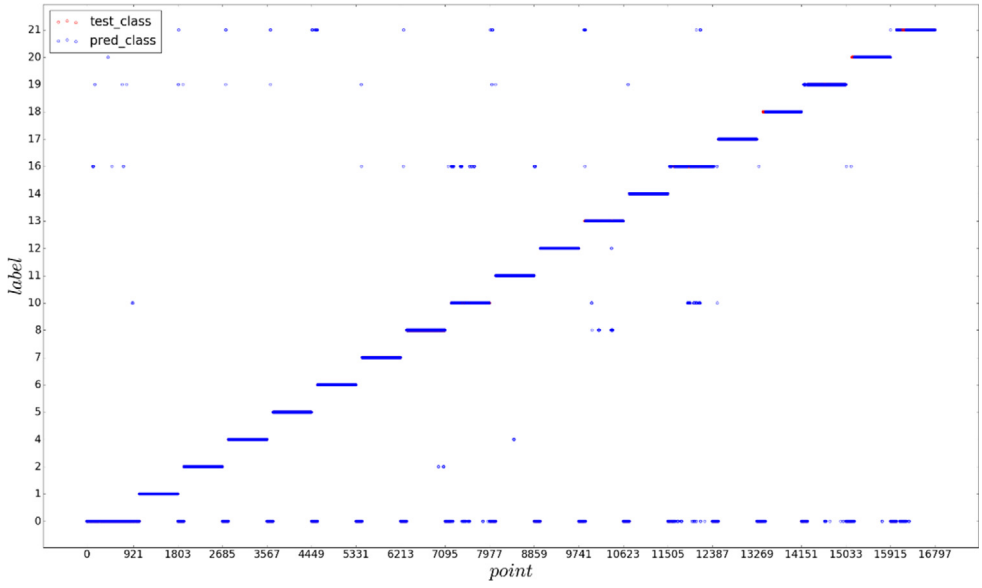

Fig. 9. Fault classification results of EDBN-2 with 19-category. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5. Concluding remarks

A novel extended DBN is designed for feature representation and fault diagnosis in chemical processes in this paper. Since most deep learning models do not consider the loss of potential valuable information in the raw data caused by layer-wise feature compressing, EDBN is developed to alleviate this problem. The strategy of repeatedly stacking raw data is used in the pre-training phase of EDBN to adequately extract useful information. By combining the previous hidden features with the original observed data as the inputs of the next ERBM, layer-wise raw data related feature can be extracted. Then, a dynamic EDBN-based fault classification modeling framework is built to consider the dynamics of process data. TE benchmark is used to assess the fault classification performance of the extended DBN. By testing

Please cite this article as: Y. Wang, Z. Pan, X. Yuan et al., A novel deep learning based fault diagnosis approach for chemical process with extended deep belief network. ISA Transactions (2019), https://doi.org/10.1016/j.isatra.2019.07.001.

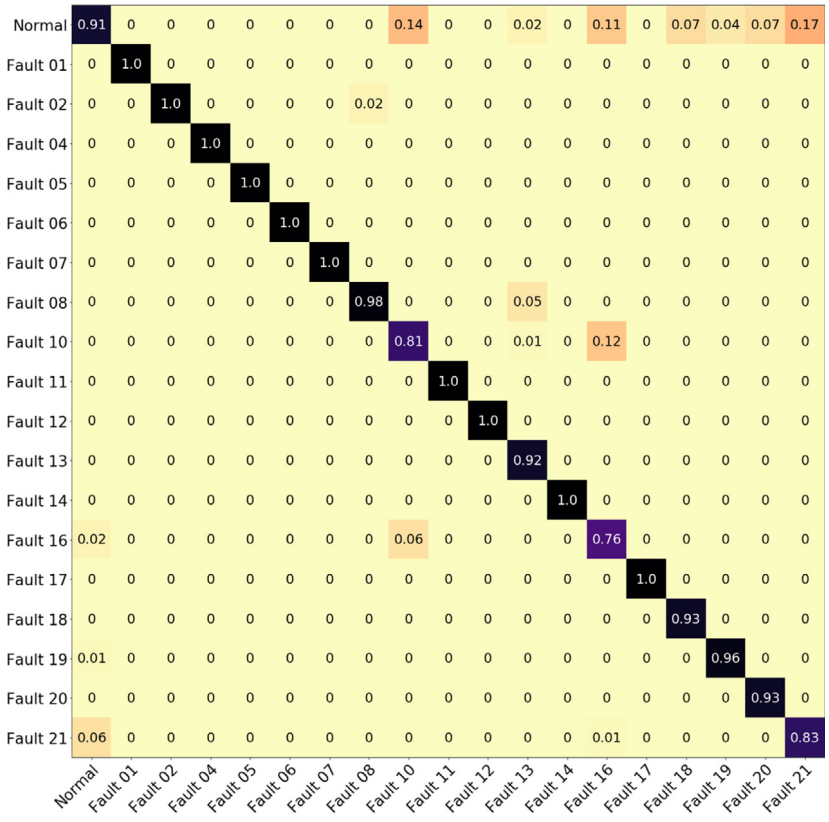

Fig. 10. The details of the proportion of samples in each fault divided into other category.

the classification performance of DBN and EDBN with different network structures on the 19-category fault datasets, it shows that EDBN can almost obtain higher FDR and lower FPR than DBN, whether it is on a single category or whole data set. In the best EDBN model, the average fault diagnosis rate achieves 94.31%, which is increased by 0.42% compared with DBN. Therefore, EDBN can extract more valuable features from raw data for further fault classification performance than the original DBN, which shows great potential for fault diagnosis in chemical processes.

Acknowledgments

This paper is supported in part by the National Natural Science Foundation of China (61590921, 61703440, 61621062, 61860206014), and 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, 2019zzts274).

Declaration of competing interest

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

References

[1] Wu H, Zhao JS. Deep convolutional neural network model based chemical process fault diagnosis. Comput Chem Eng 2018;115:185–97.
[2] Castro MAL, Escobar RF, Torres L, Aguilar JFG, Hernández JA, Olivares-Peregrino VH. Sensor fault detection and isolation system for a condensation process. ISA Trans 2016;65:456–67.
[3] Liu Y, Bazzi AM. A review and comparison of fault detection and diagnosis methods for squirrel-cage induction motors: State of the art. ISA Trans 2017;70:400–9.
[4] Yuan X, Li L, Wang Y. Nonlinear dynamic soft sensor modeling with supervised long short-term memory network. IEEE Trans Ind Inf 2019. http://dx.doi.org/10.1109/TII.2019.2902129.
[5] Wang S, Ren X, Na J, Zeng T. Extended-state-observer-based funnel control for nonlinear servomechanisms with prescribed tracking performance. IEEE Trans Autom Sci Eng 2017;14:98–108.
[6] Liu L, Liu Y, Tong S. Fuzzy based multi-error constraint control for switched nonlinear systems and its applications. IEEE Trans Fuzzy Syst 2018;1.
[7] Yuan X, Ge Z, Huang B, Song Z. A probabilistic just-in-time learning framework for soft sensor development with missing data. IEEE Trans Control Syst Technol 2017;25:1124–32.
[8] Chen N, Dai J, Yuan X, Gui W, Ren W, Koivo HN. Temperature prediction model for Roller Kiln by ALD-based double locally weighted kernel principal component regression. IEEE Trans Instrum Meas 2018;67:2001–10.
[9] Yuan X, Chen Z, Wang Y. Probabilistic nonlinear soft sensor modeling based on generative topographic mapping regression. IEEE Access 2018;6:10445–52.
[10] Yuan X, Wang Y, Yang C, Ge Z, Song Z, Gui W. Weighted linear dynamic system for feature representation and soft sensor application in nonlinear dynamic industrial processes. IEEE Trans Ind Electron 2018;65:1508–17.
[11] Zhang ZP, Zhao JS. A deep belief network based fault diagnosis model for complex chemical processes. Comput Chem Eng 2017;107:395–407.
[12] Ge ZQ. Process data analytics via probabilistic latent variable models: A tutorial review. Ind Eng Chem Res 2018;57:12646–61.
[13] Chen ZW, Ding SX, Zhang K, Li ZB, Hu ZK. Canonical correlation analysis-based fault detection methods with application to alumina evaporation process. Control Eng Pract 2016;46:51–8.
[14] Chen Z, Ding SX, Peng T, Yang C, Gui W. Fault detection for non-Gaussian processes using generalized canonical correlation analysis and randomized algorithms. IEEE Trans Ind Electron 2018;65:1559–67.
[15] Hu Z, Chen Z, Gui W, Jiang B. Adaptive PCA based fault diagnosis scheme in imperial smelting process. ISA Trans 2014;53:1446–55.
[16] Jiang QC, Yan XF, Huang BA. Performance-driven distributed, PCA process monitoring based on fault-relevant variable selection and Bayesian inference. IEEE Trans Ind Electron 2016;63:377–86.
[17] Yuan X, Ge Z, Huang B, Song Z, Wang Y. Semisupervised JIL framework for nonlinear industrial soft sensing based on locally semisupervised weighted pcr. IEEE Trans Ind Inf 2017;13:532–41.
[18] Chiang LH, Kotanchek ME, Kordon AK. Fault diagnosis based on Fisher discriminant analysis and support vector machines. Comput Chem Eng 2004;28:1389–401.

[19] Yuan X, Zhou J, Wang Y, Yang C. Multi-similarity measurement driven ensemble just-in-time learning for soft sensing of industrial processes. J Chemom 2018;32. e3040.
[20] Yuan X, Ge Z, Song Z. Locally weighted kernel principal component regression model for soft sensing of nonlinear time-variant processes. Ind Eng Chem Res 2014;53:13736–49.
[21] Jiao J, Zhao N, Wang G, Yin S. A nonlinear quality-related fault detection approach based on modified kernel partial least squares. ISA Trans 2017;66:275–83.
[22] Shen Y, Wu ZG, Shi P, Su HY, Huang TW. Asynchronous filtering for Markov jump neural networks with quantized outputs. IEEE Trans Syst Man Cybern A 2019;49:433–43.
[23] Dong SL, Wu ZG, Shi P, Karimi HR, Su HY. Networked fault detection for Markov jump nonlinear systems. IEEE Trans Fuzzy Syst 2018;26:3368–78.
[24] Ben Salem S, Bacha K, Chaari A. Support vector machine based decision for mechanical fault condition monitoring in induction motor using an advanced Hilbert-park transform. ISA Trans 2012;51:566–72.
[25] Yuan XF, Huang B, Wang YL, Yang CH, Gui WH. Deep learning-based feature representation and its application for soft sensor modeling with variable-wise weighted SAE. IEEE Trans Ind Inf 2018;14:3235–43.
[26] Liu Y, Yang C, Gao Z, Yao Y. Ensemble deep kernel learning with application to quality prediction in industrial polymerization processes. Chemometr Intell Lab Syst 2018;174:15–21.
[27] Xuan Q, Fang B, Liu Y, Wang J, Zhang J, Zheng Y, et al. Automatic pearl classification machine based on a multistream convolutional neural network. IEEE Trans Ind Electron 2018;65:6538–47.
[28] Liu Y, Fan Y, Chen J. Flame images for oxygen content prediction of combustion systems using DBN. Energy Fuels 2017;31:8776–83.
[29] Xuan Q, Chen Z, Liu Y, Huang H, Bao G, Zhang D. Multi-view generative adversarial network and its application in pearl classification. IEEE Trans Ind Electron 2018;1.
[30] Yuan X, Ou C, Wang Y, Yang C, Gui W. Deep quality-related feature extraction for soft sensing modeling: A deep learning approach with hybrid VW-SAE. Neurocomputing 2019. http://dx.doi.org/10.1016/j.neucom.2018.11.107.
[31] Jiang P, Hu ZX, Liu J, Yu SN, Wu F. Fault diagnosis based on chemical sensor data with an active deep neural network. Sensors 2016;16.
[32] Jiang L, Ge ZQ, Song ZH. Semi-supervised fault classification based on dynamic sparse stacked auto-encoders model. Chemometr Intell Lab Syst 2017;168:72–83.
[33] Yu JB, Yan XF. Layer-by-layer enhancement strategy of favorable features of the deep belief network for industrial process monitoring. Ind Eng Chem Res 2018;57:15479–90.
[34] Nair V, Hinton GE. Rectified linear units improve restricted Boltzmann machines. In: Proceedings of the 27th international conference on international conference on machine learning. Haifa, Israel: Omnipress; 2010, p. 807–14.
[35] Shang C, Yang F, Huang DX, Lyu WX. Data-driven soft sensor development based on deep learning technique. J Process Control 2014;24:223–33.
[36] Hinton GE. Training products of experts by minimizing contrastive divergence. Neural Comput 2002;14:1771–800.
[37] Hinton GE. A practical guide to training restricted Boltzmann machines. In: Montavon G, Orr GB, Müller K-R, editors. Neural networks: tricks of the trade. second ed.. Berlin, Heidelberg: Springer Berlin Heidelberg; 2012, p. 599–619.
[38] Hinton GE, Osindero S, Teh YW. A fast learning algorithm for deep belief nets. Neural Comput 2006;18:1527–54.
[39] Carreira-Perpinan MA, Hinton GE. On contrastive divergence learning. Aistats 2005;33–40.
[40] Fischer A, Igel C. An introduction to restricted Boltzmann machines. Berlin, Heidelberg: Springer Berlin Heidelberg; 2012, p. 14–36.
[41] Kingma DP, Ba J. Adam: A method for stochastic optimization. arXiv: 14126980. 2014.
[42] Yin S, Ding SX, Haghani A, Hao HY, Zhang P. A comparison study of basic data-driven fault diagnosis and process monitoring methods on the benchmark Tennessee Eastman process. J Process Control 2012;22:1567–81.
[43] Chiang LH, Russell EL, Braatz RD. Fault detection and diagnosis in industrial systems. Springer; 2000.

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 is currently a Professor. Her research interests include the modeling, optimization and control for complex industrial processes, intelligent control, and process simulation.

Zhuofu Pan is currently a Ph.D. student at the School of Information Science and Engineering, Central South University, Changsha, China. He received his the B.Eng. degree in School of Civil Engineering from Changsha University of Science & Technology and the M.Eng. degree in School of Civil Engineering from Central South University, Changsha, China, in 2017 and 2014, respectively. His research interests include deep learning and artificial intelligence, industrial big data analysis, fault detection and diagnosis, intelligent optimization algorithm, etc.

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 Information Science and Engineering, 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.

Chunhua Yang received the M.Eng. degree in automatic control engineering and the Ph.D. degree in control science and engineering from Central South University, Changsha, China, in 1988 and 2002, respectively. She was with the Department of Electrical Engineering, Katholieke Universiteit Leuven, Leuven, Belgium, from 1999 to 2001. She is currently a Full Professor with Central South University. Her current research interests include modeling and optimal control of complex industrial process, intelligent control system, and fault-tolerant computing of real-time systems.

Weihua Gui received the degree of the B.Eng. and the M.Eng. at Department of Control Science and Engineering from Central South University, Changsha, China, in 1976 and 1981, respectively. From 1986 to 1988 he was a visiting scholar at Universit-GH-Duisburg, Germany. He has been a full professor in the School of Information Science & Engineering, Central South University, Changsha, China, since 1991. His main research interests include modeling and optimal control of complex industrial processes, distributed robustcontrol, and fault diagnoses. He was elected as an academician of Chinese Academy of Engineering in 2013.

Please cite this article as: Y. Wang, Z. Pan, X. Yuan et al., A novel deep learning based fault diagnosis approach for chemical process with extended deep belief network. ISA Transactions (2019), https://doi.org/10.1016/j.isatra.2019.07.001.