Control Engineering Practice 104 (2020) 104614

Contents lists available at ScienceDirect

Control Engineering Practice

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

A dynamic CNN for nonlinear dynamic feature learning in soft sensor modeling of industrial process data

Xiaofeng Yuan, Shuaibin Qi, Yalin Wang [*], Haibing Xia

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

ARTICLE INFO

Keywords:
Deep learning
Convolutional Neural Network (CNN)
Soft sensor
Quality prediction

ABSTRACT

Hierarchical local nonlinear dynamic feature learning is of great importance for soft sensor modeling in process industry. Convolutional neural network (CNN) is an excellent local feature extractor that is suitable for process data representation. In this paper, a dynamic CNN (DCNN) strategy is designed to learn hierarchical local nonlinear dynamic features for soft sensor modeling. In DCNN, each 1D process sample is dynamically augmented into 2D data sample with lagged unlabeled process variables, which contains both spatial cross-correlations and temporal auto-correlations. Then, the convolutional and pooling layers are alternately utilized to extract the local nonlinear spatial–temporal feature from the 2D sample data matrix. Moreover, the principle is analyzed for DCNN on how it can learn the local nonlinear spatial–temporal feature from the network. The effectiveness of proposed DCNN is verified on an industrial hydrocracking process.

1. Introduction

In modern complex industrial plants, it is indispensable to implement advanced process control, optimization and monitoring techniques to ensure production safety, improve product quality and meet environmental policies (Khatibisepehr, Huang, & Khare, 2013; Sun, Jianbin, Karimi, & Fu, 2020; Sun, Liu, Qiu, & Feng, 2020; Sun, Qiu, Karimi, & Gao, 2019). The real-time feedback of process quality variables like chemical compositions and product properties is essential for these advanced techniques. On one hand, many of the quality variables are difficult to measure online due to technical and economic limitations like severe measuring environment, large measurement delays, expensive costs, etc. On the other hand, there are many routine process variables like temperatures, pressures, liquid levels and flowrates, that have strong correlations with these quality variables. Therefore, soft sensors have been developed to provide frequent online estimations for the difficult-to-measure quality variables through those easy-to-measure process variables by building predictive inferential models between them (Ge, 2018; Wang & Zhao, 2020; Yuan, Ou, Wang, Yang, & Gui, 2020).

During the last decades, many different methods have been developed for soft sensor modeling. Generally, these methods can be mainly classified into two categories, which are first-principle models (FPM) (Huang, Qi, & Murshed, 2013) and data-driven models (DDM) (Kadlec, Gabrys, & Strandt, 2009). FPMs are usually developed based on specific process physical and chemical backgrounds. However, it is usually very laborious and time-consuming to obtain accurate process physiochemical knowledge, which limits the applicability and flexibility of FPMs across different processes in real-life industrial scenarios. With the wide use of distributed control system, a large number of running data can be sampled and stored from the industrial processes. The massive historical data contains significant process information for data-driven soft sensors. Therefore, data-driven soft sensors have been extensively developed and largely researched with the rich data resources during the past decades. Many data-driven soft sensor methods, like principal component regression (Bao, Zhu, Du, Zhong, & Qian, 2019), partial least squares (Ma, Khatibisepehr, & Huang, 2015), and support vector regression (Yi & Chen, 2013), have been successfully applied to petrochemical, metallurgical, biochemical and pharmaceutical industries (Mei, Su, Liu, Ding, & Liao, 2017; Yuan, Ou, Wang, Yang, & Gui, 2019; Zhu, Ge, & Song, 2018).

Due to the large and redundant installation of measuring devices for process control and monitoring, process data usually has strong correlations, high redundancies and nonlinearities (Yan & Yan, 2019). To reduce model computational cost, enhance model robustness and improve prediction performance, it is of great significance to carry out feature learning to capture the underlying driving causality to represent the raw data. During the past decades, different feature extractors, like principal component analysis (Chen et al., 2018) and partial least squares (Hazama & Kano, 2015), have been applied for feature representation of process data in many industrial plants. Moreover, they have been extended to their nonlinear counterparts to extract more complex feature types (Dai, Chen, Yuan, Gui, & Luo, 2020). Despite these approaches have provided good performance in some situations,

* Corresponding author.
E-mail addresses: yuanxf@csu.edu.cn (X. Yuan), ylwang@csu.edu.cn (Y. Wang).

https://doi.org/10.1016/j.conengprac.2020.104614
Received 21 March 2020; Received in revised form 21 August 2020; Accepted 22 August 2020
Available online 28 August 2020

they are still very limited in their feature expressive ability for complicated data patterns in large-scale processes since they are mostly shallow networks with no more than one hidden layer.

Deep learning is a recently developed revolutionary technique for hierarchical feature representation. After it was proposed in 2006, it has successfully achieved breakthrough results and outperformed many traditional state-of-the-art machine learning models on different tasks (Hinton, Osindero, & Teh, 2006). By far, deep learning has rapidly raised great popularity and attention in different areas. Deep networks usually consist of many stacked nonlinear layers, which are able to learn more abstract features from concrete ones layer by layer. There are many different deep learning networks, like stacked autoencoder (SAE) (Wang, Yang, et al., 2020; Yu, Hong, Rui, & Tao, 2017), deep belief network (DBN) (Le Roux & Bengio, 2008; Wang, Pan, Yuan, Yang, & Gui, 2020), recurrent neural network (RNN) (Greff, Srivastava, Koutnik, Steunebrink, & Schmidhuber, 2015; Loy-Benitez, Heo, & Yoo, 2020) and convolutional neural network (CNN) (He, Zhang, Ren, & Sun, 2016; Szegedy et al., 2015). With multi-layer nonlinear transformations, deep learning is able to extract hierarchical nonlinear data features, which is very suitable for process data modeling of soft sensor (Shao, Ge, Song, & Wang, 2019; Shen, Yao, & Ge, 2020; Yan, Wang, & Jiang, 2020). For example, DBN was first applied for soft sensing of a crude distillation unit to estimate 95% cut point of the heavy diesel, in which deep learning can yield better representation ability (Shang, Yang, Huang, & Lyu, 2014). Also, a semi-supervised deep learning model was constructed for soft sensor with hierarchical extreme learning machine based autoencoder (ELM-AE), which can utilize large number of unlabeled data to improve the prediction accuracy (Yao & Ge, 2017). A stacked quality-driven autoencoder (Yuan, Zhou, et al., 2020) was proposed to extract hierarchical quality-relevant features for regression prediction.

As can be seen, the aforementioned deep learning approaches used for soft sensor are mainly based on DBN and SAE, which have powerful representation for complex nonlinear data patterns. However, most of them are static models to deal with spatial variable correlations, in which temporal data correlations are not considered. That is to say, the process samples are assumed to be temporally uncorrelated, independently and identically distributed in these models. As a matter of fact, most industrial processes are naturally dynamic as a result of process reaction kinetics, implementation of feedback control, correlated noises, dynamic disturbances, etc. There are strong temporal dependences between adjacent samples. To deal with temporal relationships, RNN and its variants like LSTM (Yuan, Li, Shardt, Wang, & Yang, 2020) have been developed for dynamic soft sensor modeling. On one hand, in these RNN-based models, the data samples must be sampled with an equal and fixed interval for the input and output variables. In this way, the adjacent data samples can obey the uniform mathematical equation in RNNs. This requirement is sometimes unpractical in many process industries, in which the quality variable is sampled and analyzed by off-line laboratory analyzer with unequal frequency. On the other hand, SAE, DBN, RNN, and LSTM are fully connected networks. They are global feature extractors in which the local data patterns are difficult to obtain. For most industries, process variables usually have strong local behaviors due to their contiguous topology structures and physicochemical relationships. For example, in the hydrocracking process, the tower top temperature and pressure often have strong local relationship at the desulfurization hydrogen stripper. However, such local data patterns may be not well captured with these global feature extractors. Moreover, the unlabeled samples with only process variables are much more than the labeled samples with both process and quality variables, which is mainly because of the high-sampling frequency for process variables and the offline low-sampling frequency for quality variables. Hence, it is a basic and important problem on how to deal with the unequal sampling frequency of quality variable and extract the local nonlinear spatial–temporal feature patterns from the massive unlabeled process variables for accurate soft sensor modeling.

Alternatively, CNN is a deep local feature extractor that is composed of multi-layers of convolution and pooling operators with local patches. CNN was proposed in the late 1980s to process data of multiple arrays (Le Cun et al., 1989). It pushes deep learning to a new tide when the AlexNet won the champion of the famous ImageNet competition in 2012, which is mainly based on CNN (Krizhevsky, Sutskever, & Hinton, 2012). Since then, CNN has become the main approach for image understanding in most object recognition and computer vision problems. Hence, it is more suitable to deal with 2D data samples. For 1-dimensional (1D) signal data, 1D CNN (Abdeljaber, Avci, Kiranyaz, Gabbouj, & Inman, 2017; Li, Zhang, Zhang, & Wei, 2017) has also been developed to handle such data type like vibration acceleration signals and electrocardiogram signals. However, 1D CNN that convolutes across the time direction cannot extract the local feature between the variables in spatial dimension. For measurements from multiple sensors, 2-dimensional (2D) CNN (Khodabandehlou, Pekcan, & Fadali, 2019) could be constructed in such occasion.

In recent years, 2D CNN has also been used for dynamic modeling in process industries. For example, Wu et al. proposed 2D CNN-based fault diagnosis model for chemical process (Wu & Zhao, 2018). Two CNN-based models were proposed to develop soft sensors for CO_2 concentration prediction in the Shell coal gasification process (Wang et al., 2019). Yuan et al. proposed multichannel CNN for various local dynamic feature representation in soft sensor (Yuan, Qi, et al., 2020). In these methods, a sliding window was used to include the past data samples to construct 2D input matrix for CNN model. Though CNN was adopted to construct the dynamic models for industrial processes in these works, they did not explain why CNN is beneficial for dynamic process modeling. In another word, the principle was unknown about what kind of features can be learned by applying CNN on such data matrix. Moreover, the CNN applications only slide their convolution kernel and pooling window along the temporal direction. The widths of convolution kernels and pooling window were usually equal to the number of secondary variables. In this way, CNN learn the dynamic features for all the process variables from a global viewpoint, which is limited in learning the local dynamic pattern between subspace variables. Also, the unequal sampling frequency of the quality variable is not motivated in these existing works, which is a very common phenomenon in industrial plants like the hydrocracking process.

To dealing with the aforementioned problems, 2D CNN is further investigated for local spatial–temporal patterns learning in this paper. Thus, a dynamic CNN (DCNN) modeling framework is built for hierarchical nonlinear dynamic feature learning of process data. The main step is to first augment the 1D data vector into 2D data matrix for each sample based on the “time lag shift” technique. Usually, there are a mass of unlabeled process variables measured using hard sensors, which can be used as time-lagged duplicate vectors to compose 2D data matrix for each sample. By constructing the 2D sample data matrix, two kinds of correlations can be included. One is along the variable space, which reflects the spatial cross-correlations between different variables. The other one is the variable temporal auto-correlations along the time axis. Then, to process each data sample of 2D matrix by CNN, both local nonlinear spatial and dynamic feature hierarchies can be learned from the massive unlabeled data using local patches with convolution and pooling operators layer by layer. In the developed DCNN, we have analyzed how it can extract local nonlinear spatial–temporal features with the convolution and pooling operators. Moreover, the DCNN allows the quality variable to be sampled with unequal frequency since the nonlinear dynamic features are directly extracted from the sequential neighboring unlabeled samples. Finally, the developed DCNN-based soft sensor model is applied to an industrial hydrocracking process.

The rest of the paper is structured as follows. In Section 2, the CNN is first briefly revisited. Section 3 introduces the dynamic CNN for soft sensor, which includes how the industrial process sample data vector is augmented to form a sample data matrix and how spatial–temporal features can be learned. Then, case study is carried out to validate the effectiveness of the proposed method on an industrial hydrocracking process in Section 4. At last, conclusions are given in Section 5.

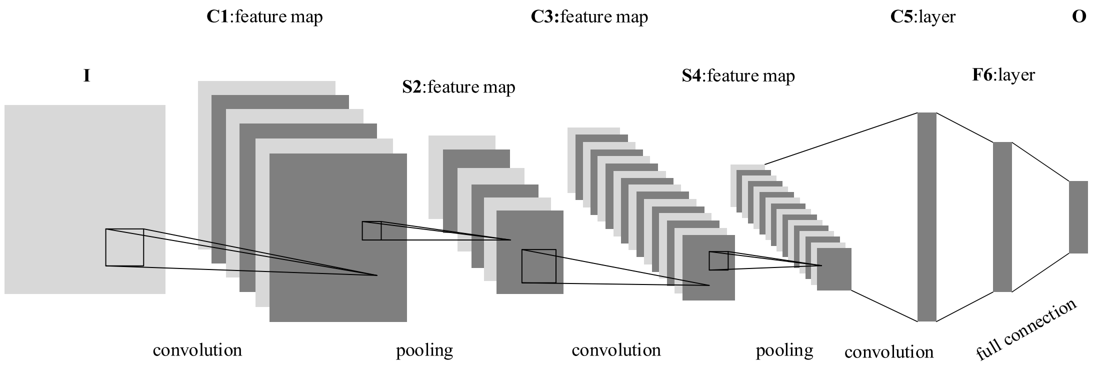

Fig. 1. The structure of the CNN.

2. Convolutional neural network

Convolutional neural network (CNN) is a deep multilayer feedforward neural network. It is widely used for image recognition since it can automatically learn complex features from the original image data and has strong fault tolerance for data noise, deformation and distortion. Moreover, CNN is very suitable for parallel and distributed computation, which can largely reduce the running time for complex tasks in the era of big data. The main idea of CNN is to greatly reduce the parameter scale of the network model via methods such as local connection, weight sharing and down sampling (or pooling). The local connection means that each convolution neuron in the convolutional layer is connected to the corresponding local sensory area of the previous layer network, and thus extracts the features of the part by a convolution kernel, which conforms to the sparse response characteristic of biological neuron. The weight sharing refers to that the connection weights between convolution neurons of the same feature map in the convolutional layer and the corresponding local sensory area of the previous layer network are same. That is to say, the same convolution kernel is used to extract the same feature type in different positions of the previous layer network.

Fig. 1. shows the structure of the CNN. As shown in Fig. 1, CNN mainly contains the input layer (I), convolutional layer (C), sampling layer (S), and output layer (O). Among them, the O layer is full-connection mode of a general feedforward network. The C layer and S layer appear alternately as the middle layers. The following is a brief introduction to the basic characteristics of each layer type of CNN network.

Input (I) layer:

The I layer of CNN network is usually a form of two-dimensional data matrix, which is characterized by the ability to automatically extract features from the original image data and realize intelligent learning without artificial participation in selection or design of appropriate features as inputs.

Convolutional (C) layer:

The C layer of CNN network is a feature extraction layer. Each convolution neuron of the feature map in C layer is connected to the corresponding local sensory area of the previous layer network, and the feature is extracted by the same convolution kernel. The size of convolution kernel is determined by the size of local connection area. The weight value constitutes the convolution kernel and the weight sharing can be realized by using the same convolution kernel to slide across the previous feature map. The advantage of C layer is that the scale of network parameters can be greatly reduced through local connection and weight sharing. Moreover, the Rectified Linear Unit (ReLU) is used as the activation function of C layer to increase the nonlinear representation ability of C layer. Fig. 2 shows the convolution operation intuitively.

Sampling (S) layer:

The S layer of CNN network, also known as the pooling layer, is often applied after the C layer. The S layer is connected to the

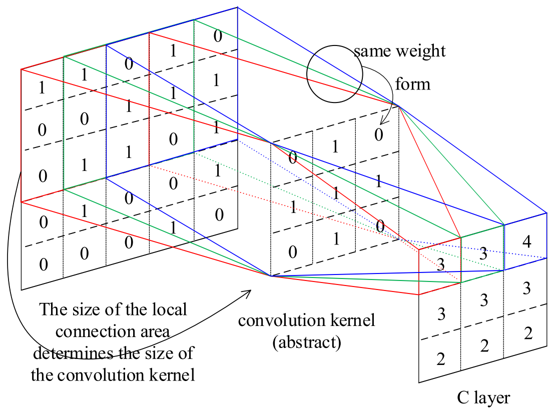
same weight
form
The size of the local connection area determines the size of the convolution kernel
convolution kernel (abstract)
C layer

Fig. 2. The schematic diagram of convolution operation.

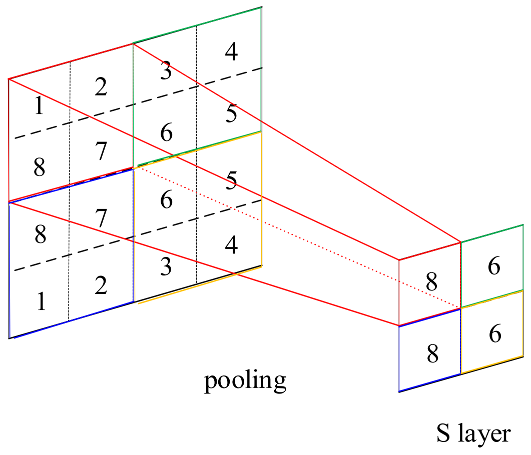

Fig. 3. The schematic diagram of maximum pooling operation.

corresponding local sensory area of the output from the previous C layer to further simplify the feature information. Unlike the C layer, the S layer takes a specific fixed operation to obtain features in the local connection area, which will not change during the training stage. The S layer not only generates no additional network parameters, but also pools the features extracted from the previous layer to further condense them. The most common pooling operations include the maximum pooling, average pooling, norm pooling and logarithmic probability pooling. For example, the maximum pooling just takes the maximum value in the local region as its output. Fig. 3 shows an example of the maximum pooling operation on a region of 2×2 neurons intuitively.

Output (O) layer:

The O layer of the CNN network is connected to the last layer of the hidden layer via a full connection way. Thus, the O layer is also called the connection layer. The features of the previous layer are flatted into a vector as the input of O layer. The O layer is characterized by fully mining the relationship between the final extracted feature and the output. The O layer should also use suitable activation function to enhance the nonlinear representation ability. For the activation function of the final layer, the classification task often uses the Softmax function while the regression task usually uses the Sigmoid function.

3. Dynamic CNN for soft sensor modeling

In industrial process plants, there are usually many continuous real-valued process variables like temperatures, pressures and flow-rates that can be easily measured online with a high-sampling frequency. Conversely, the quality variable is usually acquired by off-line analyzers with an unequal and low sampling frequency. Thus, there are a mass of unlabeled process variables, which also have high relationship with the neighboring quality variable. Suppose a total of m process variables can be measured online. With a certain sampling frequency, these variables can be sampled and measured during process running stage. Suppose the quality variable $y(t)$ is sampled at the tth sampling instant. Correspondingly, the process variables at the tth sampling instant are $\mathbf{x}(t) = \left[x_1(t), x_2(t), \ldots, x_m(t)\right]^{\mathrm{T}}$. Fig. 4 gives the sampling procedure of variables from industrial process plants, which shows the different sampling frequency of process variables and quality variable. Most of soft sensors aim to build a predictive model $y(t) = f(\mathbf{x}(t))$ that can best approximate the observed data. However, process variables are usually strongly correlated with redundant and complex information. Hence, it is necessary to carry out feature representation for the observed input data to reduce computational cost, enhance model robustness and improve prediction accuracy. Especially, dynamic CNN model is built in this part in order to learn local dynamic and nonlinear features simultaneously.

3.1. Dynamic matrix design for data sample

CNN is very powerful in local feature learning. Local features are especially important for process data modeling. However, CNN is originally designed to deal with 2D image data. Hence, it cannot be directly used for feature learning of industrial process data. Moreover, the process dynamics are usually neglected in traditional feature extractors. To simultaneously deal with these two problems in industrial processes, a dynamic CNN model is developed for local nonlinear and dynamic feature learning. The commonly used feature extractors often obtain the low-dimensional feature data from variable vector $\mathbf{x}(t)$. For example, PCA carries out dimension reduction for feature learning as

$$\mathbf{x}(t) = \mathbf{P}\mathbf{z}(t) + \mathbf{e}(t) \tag{1}$$

where $\mathbf{P}$ is the loading matrix, the columns of which are the corresponding projection axis; $\mathbf{z}(t)$ is the score vector, which is just the obtained feature data at sampling instant t; $\mathbf{e}(t)$ is the residual term. Then, the feature data $\mathbf{z}(t)$ can be used to predict the corresponding quality variable $y(t)$. As can be seen, these similar techniques often assume the sampled data have no temporal correlations. That is to say, these data samples are independently and identically distributed. However, the process variables have strong dynamic relationships with the past ones due to material transfer and chemical reaction. Thus, it is necessary to take the process dynamics into feature learning. The learned features should contain the dynamic information of process data. Between two adjacent labeled samples, there are abundant unlabeled data with the fast sampled process variables. For each sample of the quality variable, a continuous segment of unlabeled data from its previous sampling instants is chosen to construct the time-lagged 2D dynamic data. Here, a dynamic data matrix form is designed for the tth sample with the time-lagged unlabeled data as

$$\begin{aligned}\mathbf{X}(t) &= [\mathbf{x}(t), \mathbf{x}(t-1\times f), \ldots, \mathbf{x}(t-T\times f)] \\ &= \begin{bmatrix} x_1(t), & x_1(t-1\times f), & \ldots, & x_1(t-T\times f) \\ x_2(t), & x_2(t-1\times f), & \ldots, & x_2(t-T\times f) \\ \vdots & \vdots & \cdots & \vdots \\ x_m(t), & x_m(t-1\times f), & \ldots, & x_m(t-T\times f) \end{bmatrix}\end{aligned} \tag{2}$$

where T is the time lag that is obtained by process mechanism analysis, f is the sampling frequency for process variables. Usually, the process variable data $\mathbf{x}(t)$ has the corresponding quality variable $y(t)$, but its time-lagged process variables $[\mathbf{x}(t-1\times f), \ldots, \mathbf{x}(t-T\times f)]$ are usually unlabeled. Because of the high-sampling process variables and the low-sampling quality variable, the adjacent dynamic matrices of $\mathbf{X}(t)$ and $\mathbf{X}(t-1)$ usually do not overlap. By this operation, each data sample is a two-dimensional matrix with time shifted variable vectors. Then, CNN can be easily applied on the augmented data matrix for feature learning.

3.2. Convolution and sampling on dynamic matrix

After the dynamic matrix is designed for each data sample, CNN is utilized to extract deep nonlinear dynamic features for subsequent regression modeling. The particularly important techniques are the convolution and pooling operations. In this stage, it is shown and explained how the convolution operation can learn local spatial cross-correlation and temporal auto-correlation features on the dynamic matrix data in industrial processes. Moreover, the pooling operation after each convolution one is able to merge similar local features into one and prevent the overfitting problem within certain range. For better illustration, a 2 × 2 convolution kernel is taken as an example to show the spatial–temporal feature learning ability on process data. Assume a general local patch from the dynamic data matrix is denoted as $\begin{bmatrix} x_{i-1}(j-1), & x_{i-1}(j) \\ x_i(j-1), & x_i(j) \end{bmatrix}$. Here, three representative convolution kernels are analyzed for feature learning.

Type 1: Convolution kernel with a single non-zero column

$$\begin{bmatrix} x_{i-1}(j-1), & x_{i-1}(j) \\ x_i(j-1), & x_i(j) \end{bmatrix} \xrightarrow[\textit{Convolution kernel}]{\begin{bmatrix} 1/2 & 0 \\ 1/2 & 0 \end{bmatrix}} \frac{x_{i-1}(j-1) + x_i(j-1)}{2} \tag{3}$$

For the first convolution type, the kernel matrix is with a single non-zero column like $\begin{bmatrix} 1/2 & 0 \\ 1/2 & 0 \end{bmatrix}$. With this kernel matrix on the local patch, the convolution operation is used to extract a combined features between variable x_{i-1} and x_i at the sampling time $j-1$. This is just the spatial cross-correlations of different variables at each sampling instant. Moreover, this kind of spatial variable cross-correlation is shared across different sampling instants when the local patch is sliding along the sampling time direction on the 2D data matrix. Also, it can learn the same type of feature combination for different pairs of variables if the local patch is moving along the variable direction on the 2D data matrix. For this type of convolution operation, it has similar functions with traditional static feature extractors like PCA and PLS, which seek to find a feature combination of the original variables. Therefore, this kind of convolution kernel is used to learn spatial cross-correlated features from process raw data.

Type 2: Convolution kernel matrix with a single non-zero row

$$\begin{bmatrix} x_{i-1}(j-1), & x_{i-1}(j) \\ x_i(j-1), & x_i(j) \end{bmatrix} \xrightarrow[\textit{Convolution kernel}]{\begin{bmatrix} 1/2 & 1/2 \\ 0 & 0 \end{bmatrix}} \frac{x_{i-1}(j-1) + x_{i-1}(j)}{2} \tag{4}$$

For the second convolution type, the kernel matrix is with a single non-zero row like $\begin{bmatrix} 1/2 & 1/2 \\ 0 & 0 \end{bmatrix}$. If this type of convolution kernel is operated on the local patch, it is just to extract the temporal auto-correlations of variable x_{i-1} over adjacent sampling instant $j-1$ and j. With the convolution kernel sliding along the time direction on the 2D sample data matrix, temporal auto-correlated features are learned

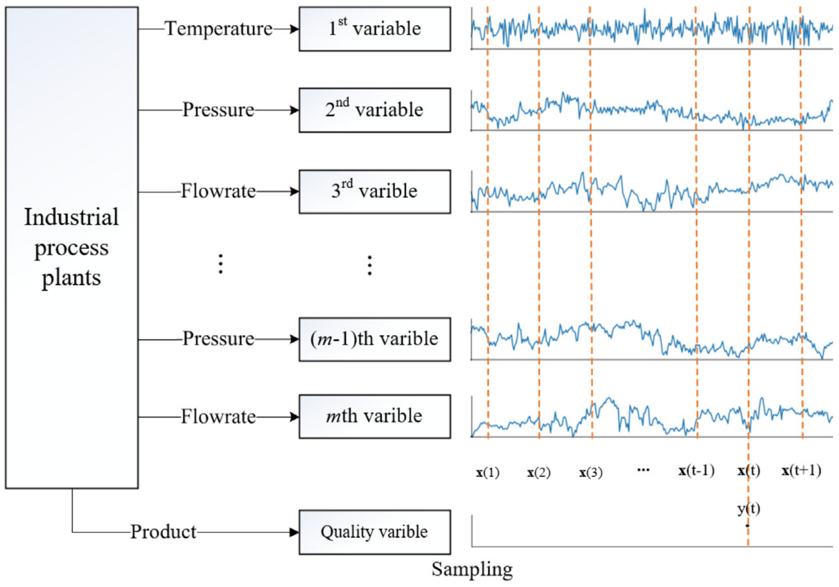

Fig. 4. The sampling procedure in industrial process plants.

for a certain variable over different adjacent sampling instants. When the operation is moving along the variable direction on the 2D data matrix, it can learn auto-correlations for different variables. Thus, this kind of convolution kernel is suitable to learn auto-correlated features from process data, which are also known as dynamic features. However, it can be easily seen that this operation on a local patch can only extract the dynamic features for one variable.

Type 3: A standard convolution kernel

$$\begin{bmatrix} x_{i-1}(j-1), & x_{i-1}(j) \\ x_i(j-1), & x_i(j) \end{bmatrix} \xrightarrow[Convolution\ kernel]{\begin{bmatrix} 1/4 & 1/4 \\ 1/4 & 1/4 \end{bmatrix}} \frac{x_{i-1}(j-1)+x_{i-1}(j)+x_i(j-1)+x_i(j)}{4} \tag{5}$$

Type 1 and 2 are two special convolution kernel matrices. Other from them, a standard convolution kernel is the one that most of the elements are non-zero in the matrix. For example, if the kernel matrix is $\begin{bmatrix} 1/4 & 1/4 \\ 1/4 & 1/4 \end{bmatrix}$, the obtained feature on the local patch is the spatial–temporal combination of variable x_{i-1} and x_i over sampling time $j-1$ and j, as shown in Eq. (5). For this type of convolution operation, both temporal auto-correlations and spatial cross-correlations variables can be learned in the local patch. Moreover, this kind of spatial–temporal features are learned for different variable combinations over different sampling durations when the convolution kernel moves across the whole 2D data matrix of each sample. After the convolution operation, there is usually a nonlinear activation function to transform the extracted the spatial–temporal features into more nonlinear representation.

Pooling (Sampling) operation

The objective of sampling layer is to merge similar local features into one. Also, the pooling operation can largely reduce the dimension of feature space since it reduces the height and width of data matrix. For a $k \times k$ pooling window, the feature dimension can be reduced to about one k^2th of the dimension of the output from the previous convolutional layer. Hence, the pooling operation can prevent the overfitting problem within certain range. Moreover, the frequently used maximum pooling can alleviate small local translations or distortions to some extent. For example, the 2D data matrix in Fig. 3 may have a slightly horizontal translation that results in each element to change its original position, just as shown in Fig. 5. However, the output data is not completely changed and the elements in the circled region in Fig. 5 are the same as the previous results after the maximum pooling operation. That is to say, the maximum pooling operation is helpful to keep the input data representation as preceding as possible when the input data translates slightly.

In such a way, high-level temporal and spatial correlations of variables can be extracted through different convolution kernels in subsequent convolutional and sampling layers.

3.3. DCNN-based soft sensor modeling

DCNN can learn high-level temporal and spatial correlated features of process data, which is suitable for soft sensor modeling. Fig. 6 shows the modeling framework of DCNN based soft sensor. The detailed procedures are described as follows:

(1). Collect raw data from industrial processes. The quality variable data are $\left[y\left(1_t\right), y\left(2_t\right), \ldots, y\left(M_t\right)\right]$ at different sampling steps, which may be sampled with unequal frequency. The process variables are collected with higher sampling frequency than the quality variables. First, outliers of the raw input data are preprocessed using the time series smoothing denoising technique based on sliding window least square method. For the preprocessed raw data, Z-score normalization is performed on the process variables to keep the samples have zero means and unit variances. Moreover, the quality variables are rescaled to [0, 1] using the Min–Max normalization.

(2) Following the sampling instant of quality variable in each training sample, augment and convert each training sample to 2D matrix $\mathbf{X}(t)$ using Eq. (2) with the unlabeled data of process variables.

(3) Determine the structure of the DCNN network, and then train it with the 2D matrix dataset of all the training samples.

(4) Augment and convert each testing sample to 2D sample data matrix using the same technique as the training stage. Obtain the prediction values with the trained DCNN model.

To train DCNN-based soft sensor model, the mean squared error (MSE) is used as the cost function of DCNN, which is defined as:

$$MSE = \frac{1}{M_t} \sum_{i=1_t}^{M_t} (y(i) - \hat{y}(i))^2 \tag{6}$$

where M_t is the number of training samples; $y(i)$ and $\hat{y}(i)$ are the true and predicted output values of the ith training sample, respectively. For soft sensor model, the smaller the cost function, the smaller the difference between y_i and $\hat{y}_i$. In this paper, the Adam algorithm is used to train the DCNN model by minimizing this cost function.

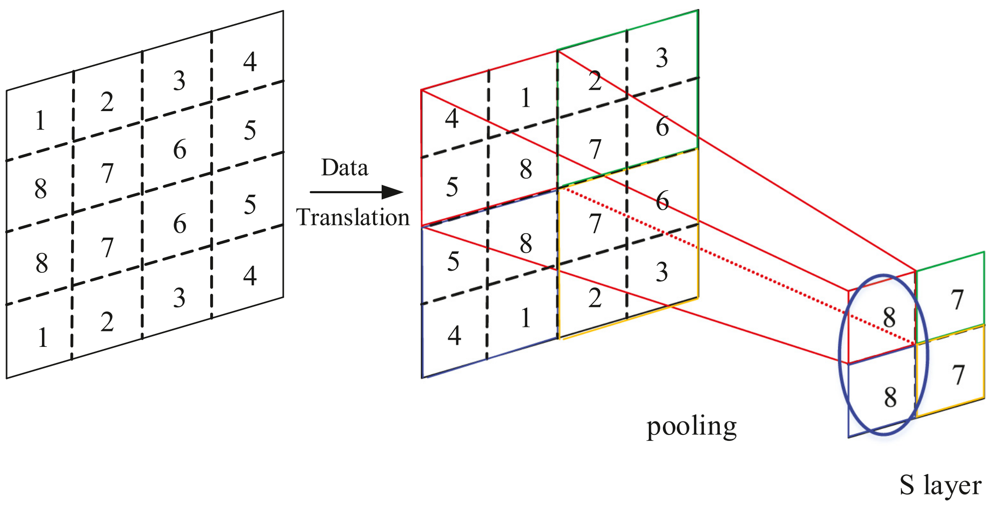

Fig. 5. The maximum pooling operation on translated input data.

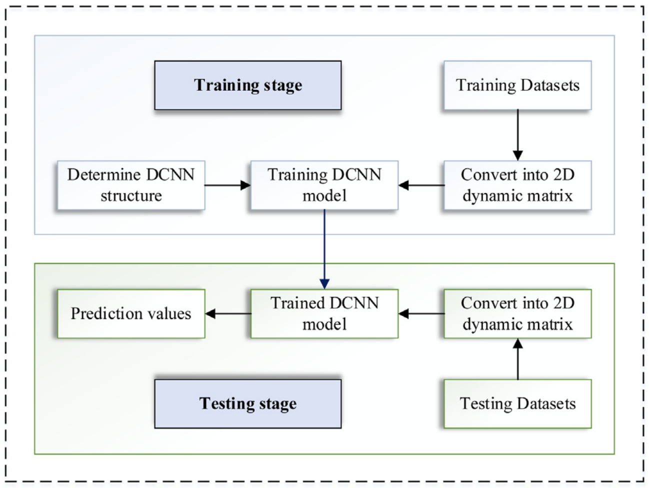

Fig. 6. The framework of DCNN based soft sensor model.

To assess the performance of the soft sensor model, root mean squared error (RMSE) and coefficient of determination (R^2) are calculated on the testing dataset, which are defined as

$$RMSE = \sqrt{\sum_{j=1_t}^{N_t} \left(y_j - \hat{y}_j\right)^2 \Big/ N_t} \tag{7}$$

$$R^2 = 1 - \sum_{j=1_t}^{N_t} (\hat{y}_j - y_j)^2 \Big/ \sum_{j=1_t}^{N_t} (y_j - \bar{y})^2 \tag{8}$$

where N_t is the number of testing samples; y_j and $\hat{y}_j$ are the true and predicted output values of the jth testing sample, respectively; $\bar{y}$ is the mean of labeled outputs in the testing dataset. RMSE can evaluate the general prediction error by the modeling methods. R^2 is a squared correlation between the actual and estimated outputs. It can provide information about how much of the total variance in the output variable data can be explained by the model.

4. Case study

In this section, the DCNN-based soft sensor is applied to an industrial hydrocracking process. Firstly, the hydrocracking process is briefly described. Then, soft sensor modeling is adopted to predict the 90% recovery temperature of the diesel oil. Besides, support vector regression (SVR), the traditional neural network (NN), DBN, and SAE are also adopted to construct the soft sensor models for performance comparison.

4.1. Description of hydrocracking process

In refining and petrochemical industry, hydrocracking process is an important unit to crack the heavy raw oils into more available light and high-quality oil products like light naphtha, kerosene and diesel. The hydrocracking process can be divided into two reaction steps: hydrogenation and cracking. The hydrogenation reaction utilizes a large amount of hydrogen to improve the hydrogen–carbon ratio of raw oils and remove impurities such as sulfur and oxygen compounds from raw oils to saturate the olefin, which releases a mass of heat to serve part of the heat source of the next cracking reaction. For the cracking reaction, carbon–carbon bond of raw oils is cracked and can also offer olefins for hydrogenation reaction. Fig. 7 gives the schematic diagram of a typical hydrocracking process. From Fig. 7, the hydrocracking process is mainly composed of four subsystems: feeding, reaction, separation and fraction subsystems. In the feeding subsystem, hydrogen and raw oils are separately heated to a certain temperature in the heating furnace and heat exchangers. Then, the raw oils and hydrogen are fed into the reaction subsystem at the same time. In the reaction subsystem, the fully mixed raw oils and hydrogen undergo the hydrogenation reaction and cracking reaction successively. The last separation and fraction subsystems mainly carry out some operations, like three phase separation, heavy-light separation, and heating–cooling fraction, to obtain the light and high-quality oil products from the effluent of the reaction subsystem.

In this complex hydrocracking process, the optimal operation conditions and running environments are some of the decisive factors for high product quality. Therefore, the real-time measurement of the operation parameters and key process indicators is critical for the adjustment and optimization of these conditions and environments. In particular, the timely information of the product properties is even more important and necessary since it can provide real-time feedback for improving the monitoring and control performance of the hydrocracking process. However, most of the product properties are obtained by off-line laboratory analyzer, which has low and unequal sampling frequency of several hours or even one day per sample. Differently, many routine process variables, such as flowrate, pressure and temperature, can be collected and measured using hard sensors with high-sampling frequency of minutes per sample. Hence, there are numerous unlabeled data of process variables in the hydrocracking

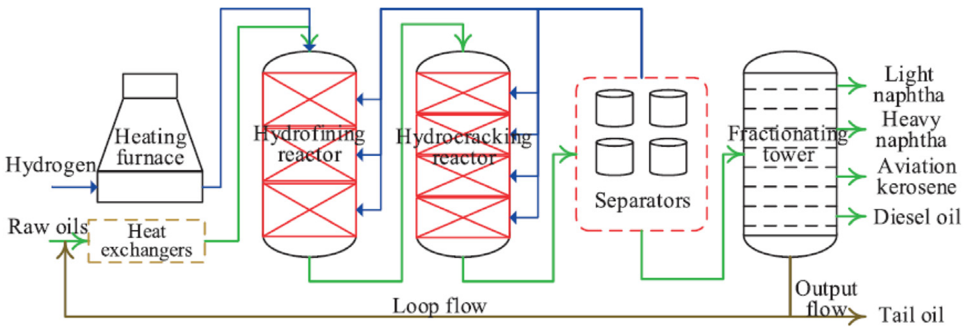

Fig. 7. The flow chat of hydrocracking process (Yuan, Zhou, Wang, & Yang, 2018).

Table 1
Descriptions of the selected 38 process variables.

Input	Variable description	Input	Variable description
1	Total feed of raw oils	20	Heat and low-pressure flow of heat and low-pressure separator
2	Total inlet temperature of heating furnace	21	Total amount of water of tank
3	Total outlet temperature of heating furnace	22	Cold and low-pressure flow of cold and low-pressure separator
4	Inlet temperature of hydrofining reactor	23	New hydrogen flow from new hydrogen compressor production to crack traffic
5	Top temperature of the first bed of hydrofining reactor	24	Return flow of desulfurization stripping tower
6	Bottom temperature of the first bed of hydrofining reactor	25	Bottom liquid flow of desulfurization stripping tower
7	Top temperature of the second bed of hydrofining reactor	26	Discharge flow from desulfurization stripping tower to debutanizer tower
8	Bottom temperature of the second bed of hydrofining reactor	27	Top pressure of main fractionating tower
9	Top temperature of the third bed of hydrofining reactor	28	Top return flow of main fractionating tower
10	Central temperature of the third bed of hydrofining reactor	29	Stream flow of stripper of main fractionating tower
11	Bottom temperature of the third bed of hydrofining reactor	30	Bottom temperature of main fractionating tower
12	Bottom temperature indication of hydrofining reactor	31	Middle output flow of main fractionating tower
13	Pressure difference of hydrofining reactor	32	Middle return flow of main fractionating tower
14	Inlet temperature of hydrocracking reactor	33	Heavy naphtha flow from heavy naphtha reflux tower to heat exchangers
15	Top temperature of the second bed of hydrocracking reactor	34	Output temperature of aviation kerosene stripper
16	Top temperature of the third bed of hydrocracking reactor	35	Output flow of aviation kerosene stripper
17	Top temperature of the fourth bed of hydrocracking reactor	36	Output temperature of diesel oil stripper
18	Pressure difference of hydrocracking reactor	37	Top temperature of diesel oil stripper
19	Top pressure of heat and low-pressure separator	38	Bottom temperature of diesel oil stripper

process, which make it possible to develop soft sensors to predict the product properties in real time. In this paper, the 90% recovery temperature of diesel oil is selected as the predicted quality variable for soft sensors modeling. Moreover, 38 process variables highly related to the quality variable are selected as the inputs of the soft sensors, the detailed descriptions of which are shown in Table 1.

4.2. Results and discussions

For the 90% recovery temperature of diesel oil, it is analyzed by off-line laboratory analyzer with unequal sampling frequency. The sampling frequency varies from four, eight or even twelve hours per sample. Totally, 1482 labeled data samples are collected from the hydrocracking process, among which 1037 (about 70%) data samples are adopted as training dataset and the remaining parts are used as testing dataset. With the sampling frequency of five minutes for the process variables, the time lag is set to be 45 by mechanism analysis and correlation calculation. Hence, each data sample can be converted into a 2D data matrix with size of 38 × 46 that contains the unlabeled data of process variables from the previous 45 sampling instants.

First, it is necessary to determine a proper network structure for constructing the DCNN-based soft sensor model. However, there are no scientific rules with good theory to design an optimal deep network structure. Usually, the trial-and-error technique is adopted to obtain a proper structure (Kalteh, 2008). In this technique, several DCNN models are constructed and compared to find a suitable one. The candidate structures are listed in Table 2. For these models, the 2D sample data matrix is served as the I layer for DCNN. Then, several pairs of C and S layers are arranged to extract features from I layer. The default stride is one for the convolution kernels. The notation of C (7 × 7 × 10) means that 10 convolution kernels with size of 7 × 7 are used in this C layer. It is similar for the other notations. All S layers use the maximum pooling with kernel size of 2 × 2 and stride of 1. The output feature maps of the last S layers are flatted to a vector as the input of the O layers. Besides, The O layers have one hidden and one output layers. For the O layers, the ReLU function and the dropout techniques (p = 0.3) are successively arranged after the input and hidden layer, which are served as activations function and regularization term, respectively. The final output layer of O layers uses one neuron and the Sigmoid function to calculate the predicted values.

The Adam algorithm is used for model training. Table 2 shows the prediction RMSE performance of these DCNN models on the testing dataset. From Table 2, model 6 has the smallest RMSE. Thus, the model 6 is chosen as the best structure for the prediction of 90% recovery temperature of diesel oil, the structure of which is shown in Fig. 8 intuitively. From Fig. 8, there are two pairs of C and S layers. In the first and second C layer, there are 14 and 20 convolution kernels with size of 7 × 7, respectively. The output feature maps of the second S layer are flatted to a vector of 1 × 700. The O layer has the network structure of [700 420 1]. The training hyperparameters are set with the learning rate of 0.001, the batch size of 64, and the epochs of 100.

To validate the effectiveness of DCNN, SVR, NN, DBN, and SAE are also adopted for soft sensor modeling. The inputs for these four models are a vector of 1748 variables (i.e., $38 \times 46 = 1748$), which is obtained by flatting the 2D input data matrix in the DCNN model. The SVR uses RBF kernel to increase nonlinear representation. The specific structures of NN, DBN, and SAE are also determined using the trial-and-error technique. Take the NN for an example, Table 3 shows some representative NN structures in the experiment, from which model 4 with the structure of [1748 420 100 1] is selected as the NN structure in this case. In the same way, The DBN is determined with the structure of [1748 300 200 100 1], and the SAE has the structure of [1748 200

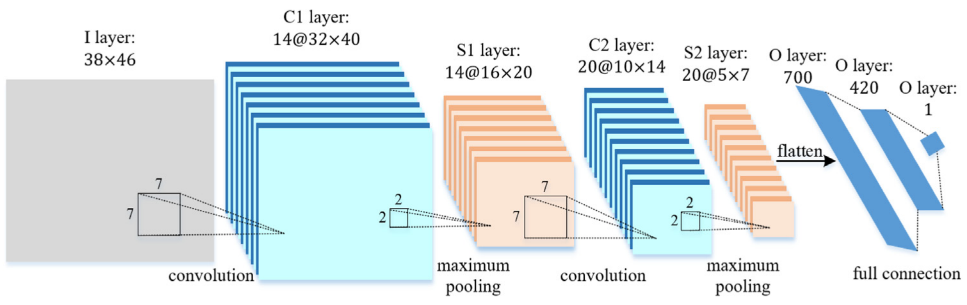

Fig. 8. Structure of DCNN for the prediction of 90% recovery temperature of diesel oil.

Table 2
DCNN model candidates for the prediction of 90% recovery temperature of diesel oil on the testing dataset.

Model	Structure	RMSE
Model 1	I (38 × 46) - C (7×7×10) - S - C (7×7×10) - S - O (350) - O (180) - O (1)	0.0917
Model 2	I (38 × 46) - C (7×7×10) - S - C (7×7×20) - S - O (700) - O (350) - O (1)	0.0887
Model 3	I (38 × 46) - C (7×7×20) - S - C (7×7×20) - S - O (700) - O (350) - O (1)	0.0833
Model 4	I (38 × 46) - C (7×7×14) - S - C (7×7×20) - S - O (700) - O (350) - O (1)	0.0846
Model 5	I (38 × 46) - C (7×7×14) - S - C (7×7×20) - S - O (700) - O (450) - O (1)	0.0845
Model 6	I (38 × 46) - C (7×7×14) - S - C (7×7×20) - S - O (700) - O (420) - O (1)	0.0820
Model 7	I (38 × 46) - C (5×5×14) - S - C (5×5×20) - S - O (960) - O (480) - O (1)	0.0874
Model 8	I (38 × 46) - C (7×7×15) - S - C (5×5×15) - S - C (5×5×20) - S - O (40) - O (20)-O (1)	0.0900
Model 9	I (38 × 46) - C (5×5×10) - S - C (5×5×15) - S - C (5×5×20) - S - O (40) - O (20) - O (1)	0.0898
Model 10	I (38 × 46) - C (3×3×20) - S - C (3×3×25) - S - C (3×3×30) - S - O (360)-O (180) - O (1)	0.0877

Table 3
NN model candidates for the prediction of 90% recovery temperature of diesel oil.

Model	Structure	RMSE
Model 1	1748–1	0.1100
Model 2	1748-874-1	0.0946
Model 3	1748-420-1	0.0926
Model 4	1748-420-100-1	0.0894
Model 5	1748-420-200-1	0.0918

Table 4
Prediction results of SVR, NN, DBN, SAE and DCNN.

Method	SVR	NN	DBN	SAE	DCNN
RMSE	0.1087	0.0894	0.0877	0.0866	0.0820
R^2	0.7108	0.8042	0.8118	0.8163	0.8352

100 10 1]. For these three compared networks, the output layer uses the Sigmoid function to scale the predicted outputs. Also, the dropout technique is arranged after the input and hidden layers. The networks are also trained with Adam algorithm by minimizing the MSE loss function in Eq. (6).

After that, Table 4 provides the RMSE and R^2 values of the five methods on the testing dataset. As can be seen, SVR provides the worst prediction performance. Unlike SVR with a shallow structure, NN, DBN, SAE, and DCNN employ multi-layer structures to describe more complex nonlinear relationship, which makes them provide better prediction performance than SVR. However, the NN has the worst performance among the four NN-based methods. DBN and SAE are firstly pre-trained in a layer-by-layer unsupervised manner to obtain the hierarchical data patterns, and then carry out fine-tuning to obtain the final model. The abstract hierarchical features are more structured for prediction task. Hence, DBN and SAE have better prediction accuracy than NN. Moreover, SVR, NN, DBN, and SAE are difficult to extract the local features since they can be regarded as global fully connect networks. Thus, these four methods only focus on the global features and the important local patterns among the input variables are neglected. For DCNN, the I layer is a 2D matrix for each sample data, which contains both spatial correlation and temporal dynamic information. Then, the local nonlinear dynamic features can be captured when the local patches (kernel) in C and S layers slide across the whole I layer. Meanwhile, the stacked C and S layers can transform the captured features into high-level abstract representations. Hence, it is more effective and suitable for soft sensor modeling. The RMSE and R^2 values of DCNN are 0.0820 and 0.8352 on the testing dataset, which reflects that DCNN has the most accurate prediction performance among the five models.

Intuitively, the detailed prediction results are depicted in Fig. 9 for the five methods on the testing dataset. Figs. 9(a), 9(b), 9(c), 9(d) and 9(e) show the predicted and true curves of the quality variable for SVR, NN, DBN, SAE and DCNN, respectively. As can be seen, SVR has the worst prediction accuracy with the large deviations, and the predicted curve of DCNN can generally track better with the true curve than the other four models. DCNN has the most accurate prediction values on most of the testing samples since it has relatively small deviations. Fig. 9(f) further provides the comparison of prediction performance with the scatter points for these five models in different colors, in which x-axis and y-axis represent the true and the predicted values, respectively. In Fig. 9(f), the purple line serves as the reference line, which represent that the predicted output values are equal to the actual ones. The closer the scatter points are to the reference line, the better the prediction performance is. It is easily seen that the scatter points of DCNN are much closer to the reference line than these of the other four models. Hence, DCNN outperforms the other four methods for prediction of the 90% recovery temperature of diesel oil.

Additionally, Fig. 10 depicts the prediction errors of SVR, NN, DBN, SAE and DCNN in two forms, which include Fig. 10(a) of the detailed prediction errors on the testing samples and Fig. 10(b) of the box plot of the prediction error. From Fig. 10(a), the prediction errors of DCNN are much closer to zero than the other methods SVR and NN on most testing samples. This also indicates that the prediction performance of DCNN is more accurate than that of the other models. Moreover, From Fig. 10(b), it can also be seen that the prediction errors of DCNN have the narrowest range around zero.

Furthermore, Table 5 gives the detailed distribution statistics of the prediction errors on the testing dataset for SVR, NN, DBN, SAE and DCNN, which include the mean of the absolute values (mean), the standard deviation (std), the minimum, 25% quantile, median, 75% quantile and maximum values. The minimum, median, maximum and quantile values listed in Table 5 are intuitively visualized in Fig. 10(b)

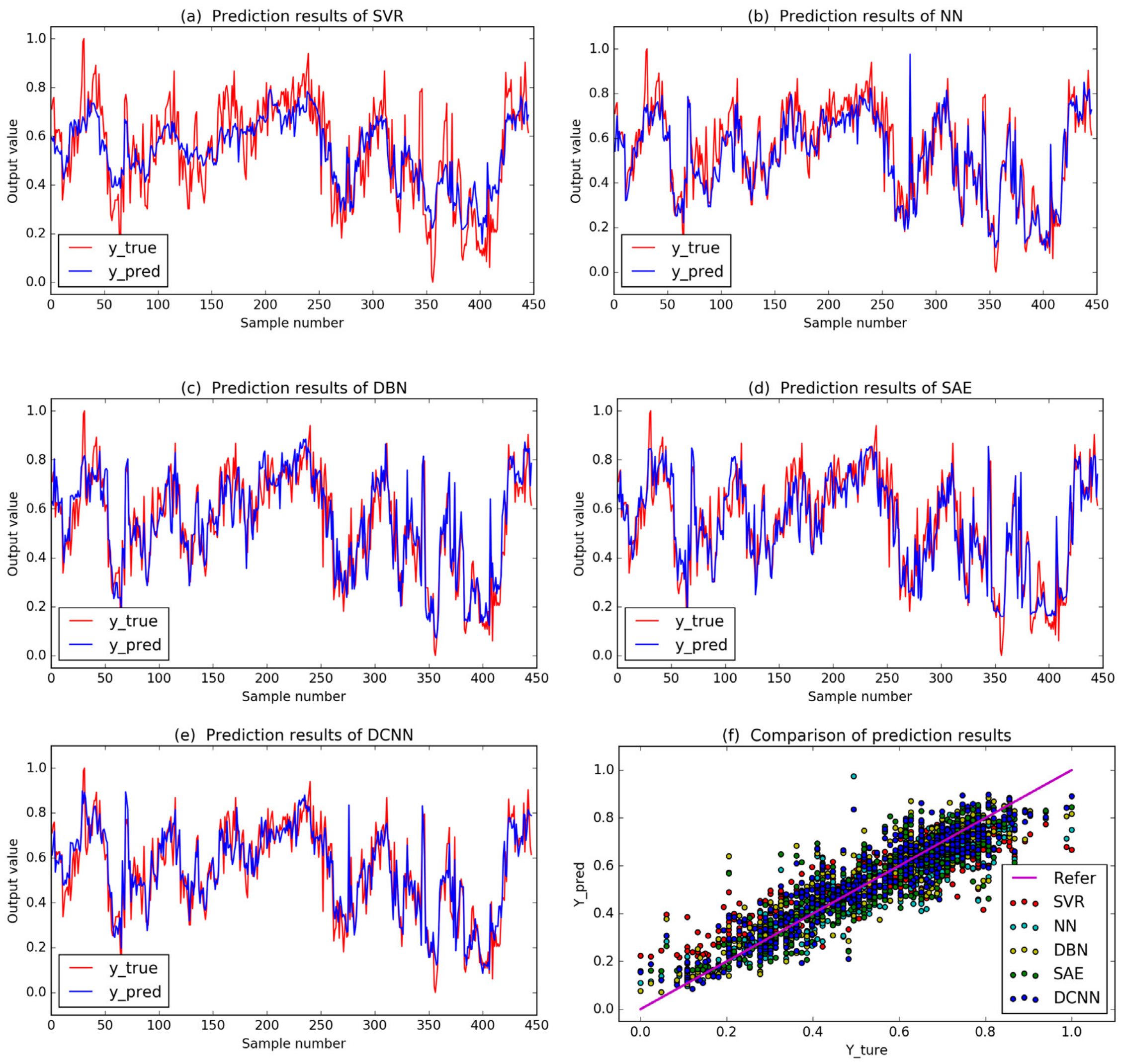

Fig. 9. The detailed prediction results of SVR, NN, DBN, SAE and DCNN on the testing dataset: (a) SVR, (b) NN, (c) DBN, (d) SAE, (e) DCNN, (f) scatter points (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

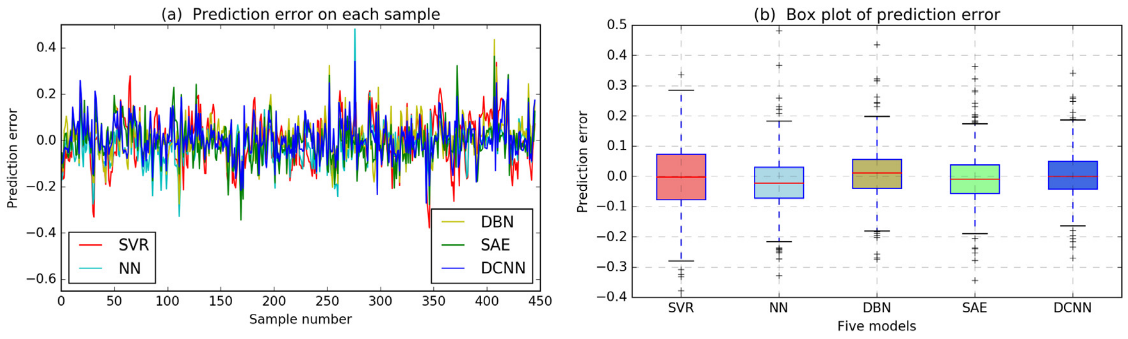

Fig. 10. The comparison of prediction errors between SVR, NN, DBN, SAE and DCNN: (a) prediction error on each sample, (b) box plot of prediction error.

of the box plots. From both Table 5 and Fig. 10(b), it can be found that the error ranges for SVR, NN, DBN, SAE and DCNN are [−0.3780, 0.3368], [−0.3276, 0.4811], [−0.2739, 0.4355], [−0.3436, 0.3638] and [−0.2711, 0.3412], respectively. That is to say, the prediction errors of DCNN are more concentrated around zero than that of the other four models. Meanwhile, the std and the mean value of the prediction errors of DCNN are the smallest among the five models. This indicates the prediction errors of DCNN oscillate very little and the prediction performance of DCNN is the most concentrated and stable. Consequently, the prediction accuracy of DCNN is improved by extracting both local nonlinear spatial and dynamic features of the input variables.

5. Conclusion

In this paper, a DCNN model is designed for hierarchical nonlinear dynamic feature representation in soft sensor application, which overcomes the limitations of most existing models. The input sample

Table 5
The distribution statistics of prediction errors.

Method	SVR	NN	DBN	SAE	DCNN
mean	0.0872	0.0669	0.0658	0.0642	0.0619
std	0.1088	0.0865	0.0872	0.0864	0.0818
minimum	−0.3780	−0.3276	−0.2739	−0.3436	−0.2711
25% quantile	−0.0775	−0.0718	−0.0385	−0.0554	−0.0412
Median	−0.0022	−0.0220	0.0120	−0.0094	0.0005
75% quantile	0.0742	0.0304	0.0568	0.0377	0.0503
maximum	0.3368	0.4811	0.4355	0.3638	0.3412

of the DCNN is a 2D data matrix augmented from 1D unlabeled sample vector with "time lag shift" technique. The variable and time axis of 2D sample matrix reflect the spatial cross-correlations and temporal auto-correlation of process variable data, respectively. By the convolutional and pooling operations on the 2D data matrix of each sample, local nonlinear dynamic feature hierarchies can be progressively learned layer by layer. An industrial application on the hydrocracking process demonstrates the effectiveness and flexibility of DCNN. From the prediction results on the testing dataset, the prediction performance of DCNN is superior to SVR, NN, DBN, and SAE. Hence, the proposed of DCNN is suitable for soft sensor modeling of industrial process data.

Declaration of competing interest

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

Acknowledgments

This work was supported in part by the Program of National Natural Science Foundation of China (61988101, 61590921, 61703440, 61621062), the National Key R&D Program of China (2018AAA0101603, 2018YFB1701100), the Natural Science Foundation of Hunan Province of China (2018JJ3687), and the Fundamental Research Funds for the Central Universities of Central South University, China (1053320191321).

References

Abdeljaber, O., Avci, O., Kiranyaz, S., Gabbouj, M., & Inman, D. J. (2017). Real-time vibration-based structural damage detection using one-dimensional convolutional neural networks. *Journal of Sound and Vibration*, *388*, 154–170.

Bao, Y., Zhu, Y., Du, W., Zhong, W., & Qian, F. (2019). A distributed PCA-TSS based soft sensor for raw meal fineness in VRM system. *Control Engineering Practice*, *90*, 38–49.

Chen, N., Dai, J., Yuan, X., Gui, W., Ren, W., & Koivo, H. N. (2018). Temperature prediction model for Roller Kiln by ALD-based double locally weighted kernel principal component regression. *IEEE Transactions on Instrumentation And Measurement*, *67*(8), 2001–2010.

Dai, J., Chen, N., Yuan, X., Gui, W., & Luo, L. (2020). Temperature prediction for roller kiln based on hybrid first-principle model and data-driven MW-DLWKPCR model. *ISA Transactions*, *98*, 403–417.

Ge, Z. (2018). Process data analytics via probabilistic latent variable models: A tutorial review. *Industrial & Engineering Chemistry Research*, *57*(38), 12646–12661.

Greff, K., Srivastava, R. K., Koutnik, J., Steunebrink, B. R., & Schmidhuber, J. (2015). LSTM: A search space Odyssey. *IEEE Transactions on Neural Networks & Learning Systems*, *28*(10), 2222–2232.

Hazama, K., & Kano, M. (2015). Covariance-based locally weighted partial least squares for high-performance adaptive modeling. *Chemometrics and Intelligent Laboratory Systems*, *146*, 55–62.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

Hinton, G. E., Osindero, S., & Teh, Y. -W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*(7), 1527–1554.

Huang, B., Qi, Y., & Murshed, A. M. (2013). *Dynamic modeling and predictive control in solid oxide fuel cells: First principle and data-based approaches*. John Wiley & Sons.

Kadlec, P., Gabrys, B., & Strandt, S. (2009). Data-driven soft sensors in the process industry. *Computers & Chemical Engineering*, *33*(4), 795–814.

Kalteh, A. M. (2008). *Rainfall-runoff modelling using artificial neural networks (ANNs): Modelling and understanding*.

Khatibisepehr, S., Huang, B., & Khare, S. (2013). Design of inferential sensors in the process industry: A review of Bayesian methods. *Journal of Process Control*, *23*(10), 1575–1596.

Khodabandehlou, H., Pekcan, G., & Fadali, M. S. (2019). Vibration-based structural condition assessment using convolution neural networks. *Structural Control and Health Monitoring*, *26*(2), Article e2308.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). *Advances in neural information processing systems* (pp. 1097–1105).

Le Cun, Y., Jackel, L. D., Boser, B., Denker, J. S., Graf, H. P., Guyon, I., et al. (1989). Handwritten digit recognition: Applications of neural network chips and automatic learning. *IEEE Communications Magazine*, *27*(11), 41–46.

Le Roux, N., & Bengio, Y. (2008). Representational power of restricted Boltzmann machines and deep belief networks. *Neural Computation*, *20*(6), 1631–1649.

Li, D., Zhang, J., Zhang, Q., & Wei, X. (2017). In *2017 IEEE 19th international conference on e-Health networking, applications and services* (pp. 1–6).

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

Ma, M., Khatibisepehr, S., & Huang, B. (2015). A Bayesian framework for real-time identification of locally weighted partial least squares. *AIChE Journal*, *61*(2), 518–529.

Mei, C., Su, Y., Liu, G., Ding, Y., & Liao, Z. (2017). Dynamic soft sensor development based on Gaussian mixture regression for fermentation processes. *Chinese Journal of Chemical Engineering*, *25*(1), 116–122.

Shang, C., Yang, F., Huang, D., & Lyu, W. (2014). Data-driven soft sensor development based on deep learning technique. *Journal of Process Control*, *24*(3), 223–233.

Shao, W., Ge, Z., Song, Z., & Wang, K. (2019). Nonlinear industrial soft sensor development based on semi-supervised probabilistic mixture of extreme learning machines. *Control Engineering Practice*, *91*, Article 104098.

Shen, B., Yao, L., & Ge, Z. (2020). Nonlinear probabilistic latent variable regression models for soft sensor application: From shallow to deep structure. *Control Engineering Practice*, *94*, Article 104198.

Sun, K., Jianbin, Q., Karimi, H. R., & Fu, Y. (2020). Event-triggered robust fuzzy adaptive finite-time control of nonlinear systems with prescribed performance. *IEEE Transactions on Fuzzy Systems*, http://dx.doi.org/10.1109/TFUZZ.2020.2979129.

Sun, K., Liu, L., Qiu, J., & Feng, G. (2020). Fuzzy adaptive finite-time fault-tolerant control for strict-feedback nonlinear systems. *IEEE Transactions on Fuzzy Systems*, http://dx.doi.org/10.1109/TFUZZ.2020.2965890.

Sun, K., Qiu, J., Karimi, H. R., & Gao, H. (2019). A novel finite-time control for nonstrict feedback saturated nonlinear systems with tracking error constraint. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. E., Anguelov, D., et al. (2015). Going deeper with convolutions. In *Procedings of the IEEE conference on computer vision and pattern recognition* (pp. 1–9).

Wang, Y., Pan, Z., Yuan, X., Yang, C., & Gui, W. (2020). A novel deep learning based fault diagnosis approach for chemical process with extended deep belief network. *ISA Transactions*, *96*, 457–467.

Wang, K., Shang, C., Liu, L., Jiang, Y., Huang, D., & Yang, F. (2019). Dynamic soft sensor development based on convolutional neural networks. *Industrial and Engineering Chemistry Research*, *58*(26), 11521–11531.

Wang, Y., Yang, H., Yuan, X., Shardt, Y., Yang, C., & Gui, W. (2020). Deep learning for fault-relevant feature extraction and fault classification with stacked supervised auto-encoder. *Journal of Process Control*, *92*, 79–89.

Wang, J., & Zhao, C. (2020). Mode-cloud data analytics based transfer learning for soft sensor of manufacturing industry with incremental learning ability. *Control Engineering Practice*, *98*, Article 104392.

Wu, H., & Zhao, J. S. (2018). Deep convolutional neural network model based chemical process fault diagnosis. *Computers & Chemical Engineering*, *115*, 185–197.

Yan, X., Wang, J., & Jiang, Q. (2020). Deep relevant representation learning for soft sensing. *Information Sciences*, *514*, 263–274.

Yan, S., & Yan, X. (2019). Using labeled autoencoder to supervise neural network combined with k-nearest neighbor for visual industrial process monitoring. *Industrial and Engineering Chemistry Research*, *58*(23), 9952–9958.

Yao, L., & Ge, Z. (2017). Deep learning of semisupervised process data with hierarchical extreme learning machine and soft sensor application. *IEEE Transactions on Industrial Electronics*, *65*(2), 1490–1498.

Yi, L., & Chen, J. (2013). Integrated soft sensor using just-in-time support vector regression and probabilistic analysis for quality prediction of multi-grade processes. *Journal of Process Control*, *23*(6), 793–804.

Yu, J., Hong, C., Rui, Y., & Tao, D. (2017). Multitask autoencoder model for recovering human poses. *IEEE Transactions on Industrial Electronics*, *65*(6), 5060–5068.

Yuan, X., Li, L., Shardt, Y. A. W., Wang, Y., & Yang, C. (2020). Deep learning with spatiotemporal attention-based LSTM for industrial soft sensor model development. *IEEE Transactions on Industrial Electronics*, http://dx.doi.org/10.1109/TIE.2020.2984443.

Yuan, X., Ou, C., Wang, Y., Yang, C., & Gui, W. (2019). A layer-wise data augmentation strategy for deep learning networks and its soft sensor application in an industrial hydrocracking process. *IEEE Transactions on Neural Networks and Learning Systems*, http://dx.doi.org/10.1109/TNNLS.2019.2951708.

Yuan, X., Ou, C., Wang, Y., Yang, C., & Gui, W. (2020). Deep quality-related feature extraction for soft sensing modeling: A deep learning approach with hybrid VW-SAE. *Neurocomputing*, *396*, 375–382.

Yuan, X., Qi, S., Shardt, Y. A. W., Wang, Y., Yang, C., & Gui, W. (2020). Soft sensor model for dynamic processes based on multichannel convolutional neural network. *Chemometrics and Intelligent Laboratory Systems*, *203*, Article 104050.

Yuan, X., Zhou, J., Huang, B., Wang, Y., Yang, C., & Gui, W. (2020). Hierarchical quality-relevant feature representation for soft sensor modeling: a novel deep learning strategy. *IEEE Transactions on Industrial Informatics*, *16*(6), 3721–3730.

Yuan, X., Zhou, J., Wang, Y., & Yang, C. (2018). Multi-similarity measurement driven ensemble just-in-time learning for soft sensing of industrial processes. *Journal of Chemometrics*, *32*(9), Article e3040.

Zhu, J., Ge, Z., & Song, Z. (2018). Quantum statistic based semi-supervised learning approach for industrial soft sensor development. *Control Engineering Practice*, *74*, 144–152.