Gated Stacked Target-Related Autoencoder: A Novel Deep Feature Extraction and Layerwise Ensemble Method for Industrial Soft Sensor Application

Qingqiang Sun, and Zhiqiang Ge, *Senior Member, IEEE*

***Abstract*—These days, data-driven soft sensors have been widely applied to estimate the difficult-to-measure quality variables in the industrial process. How to extract effective feature representations from complex process data is still the difficult and hot spot in the soft sensing application field. Deep learning (DL), which has made great progresses in many fields recently, has been used for process monitoring and quality prediction purposes for its outstanding nonlinear modeling and feature extraction abilities. In this work, deep stacked autoencoder (SAE) is introduced to construct a soft sensor model. Nevertheless, conventional SAE-based methods do not take information related to target values in the pretraining stage and just use the feature representations in the last hidden layer for final prediction. To this end, a novel gated stacked target-related autoencoder (GSTAE) is proposed for improving modeling performance in view of the above two issues. By adding prediction errors of target values into the loss function when executing a layerwise pretraining procedure, the target-related information is used to guide the feature learning process. Besides, gated neurons are utilized to control the information flow from different layers to the final output neuron that take full advantage of different levels of abstraction representations and quantify their contributions. Finally, the effectiveness and feasibility of the proposed approach are verified in two real industrial cases.**

***Index Terms*—Deep learning (DL), gated neurons, nonlinear feature extraction, soft sensor, stacked autoencoder (SAE), target-related information.**

I. Introduction

IN MODERN industrial processes, it is of great importance to monitor the quality of products and other key variables, in order to keep the process in a safe state and provide an effective control and optimization style [1], [2]. However, due to poor measuring environments and expensive analytic costs, it is hard to measure those important variables in time for process control. Therefore, soft sensing technology emerges as the times require which selects a set of measurable variables (auxiliary variables) related to those to be estimated variables (dominant variables), and then constructs a mathematical model with auxiliary variables as input and dominant variables as output to estimate those variables that cannot be measured directly [3], [4].

Generally, there are three kinds of soft sensor modeling methods, including mechanism-based models, knowledge-based models, and data-driven models [5]. With the rapid development of computer technology and the maturity of information collection, transmission, and storage technologies, as well as the widespread application of distributed control systems (DCSs), industrial processes produce and store a large amount of measurement data every day [6]. Hence, data-driven modeling methods have drawn more and more attention and been increasingly employed for soft sensor constructions [7], [8].

As a kind of typical data-driven approach, deep learning (DL) methods, developed from artificial neural-network techniques (ANN), have dramatically improved the state of the art in many fields, such as speech recognition, visual object recognition, object detection, and so on [9]–[11]. Compared with conventional methods based on the hand-design feature extractor, DL learns useful feature representation by itself without the requirement of engineering skills and domain expertise since it is an end-to-end learning approach [12], [13]. Besides, the representations extracted by DL are at multiple levels of abstraction because multiple nonlinear processing layers are stacked [14], [15]. Adopting the greedy layer-by-layer learning approach, DL effectively solves the backpropagation problem of gradients when depth increases, which gives those methods a more powerful data modeling ability in contrast with shallow representation learning [16]. To summarize DL methods have outstanding data mining and nonlinear modeling ability, which are quite suitable for soft sensors applications since the nonlinear issue exists widely in practical industrial scenarios [17]–[19]. In the industrial field, more and more methods based on DL have been applied for process monitoring and soft sensor modeling purposes [20]–[22].

Among those various models, a stacked autoencoder (SAE) is one of the most widely used network. There are many

Manuscript received April 18, 2020; accepted July 15, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant 61722310, and in part by the Natural Science Foundation of Zhejiang Province under Grant LR18F030001. This article was recommended by Associate Editor W. Wan. *(Corresponding author: Zhiqiang Ge.)*

The authors are with the State Key Laboratory of Industrial Control Technology, Institute of Industrial Process Control, College of Control Science and Engineering, Zhejiang University, Hangzhou 310027, China (e-mail: sunqingqiang@zju.edu.cn; gezhiqiang@zju.edu.cn).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2020.3010331

variants of SAE that are designed to meet specific requirements for better modeling performance. For example, a typical sparse autoencoder is used to avoid a serious overfitting problem when there are far more neurons in the hidden layer than that in the input layer [23]. The basic theory of SAE focuses on extracting useful features by minimizing the reconstruction error of input data. It can make sense because the information about the input data are preserved as fully as possible. However, this would lead to a simple reproduction of input, which is just not enough for practical applications. To help SAE learn more information, several works have been proposed. Rasmus *et al.* [24] proposed a method called a ladder network, which consists of two forward encoders and a backward decoder. The forward encoding parameters are shared between two encoders, but the input data of the auxiliary encoder are contaminated. Every decoding layer tries to reconstruct the clean latent variables from the noisy feature in the encoding layer at the same depth and the reconstructed feature in the deeper layer of the decoder through several denoising functions. By minimizing the reconstruction errors, the networks are updated. In this way, the ladder network ensures the features extracted from corrupted input has the same quality as that from clean data. However, this mainly enhances the generalization performance without improving the upper bound of the original encoder since it still focuses on reconstructing only the input or latent variables as well as possible. In other words, the feature representations learned in this way actually ignore the information related to output value which influences the performance of the subsequent supervised fine-tuning procedure. For higher predictive accuracy, the guidance of target labels is of great significance. Hence, Yuan *et al.* [25] introduced a variablewise weighted SAE (VWSAE) method for hierarchical output-related feature representation by assigning variables with different weights according to correlation analysis with the output variable during the pretraining process. Nevertheless, Pearson's correlation coefficients are calculated as the index of the correlation between latent variables and the target output, which is just a linear correlation measurement. Therefore, it is necessary to find a more suitable means to establish a connection between input/latent variables and output values in the complex data environment. Moreover, although features in different layers are representations of data at different levels of abstraction, most published works about neural networks only directly generate the predictive values using the features extracted by the last hidden layer [26], [27]. In this way, the contributions of different hidden layers to output prediction are ignored, which leads to the loss of information learned by layerwise pretraining. Therefore, what matters most is how to evaluate their contributions and control their information flows. However, to the best of our knowledge, these research aspects have not yet been investigated and discussed in the existing works.

In this article, focusing on the above two issues, a novel DL model called the gated stacked target-related autoencoder (GSTAE) is proposed for the soft sensor application. Although the entire learning process of GSTAE still includes pretraining and fine-tuning stages, both two procedures are redesigned and optimized, which validate the novelty of this work and constitute its contributions. First, the target-related information is considered in the pretraining stage by adding the prediction error of the target variable into the original loss function when feature extraction is carried out layer by layer. In this way, the multiple-layer network is pretrained in a supervised manner. Second, after the pretraining procedure, instead of using only the feature of the last layer to obtain the predicted value, gated neurons are utilized to take full advantage of feature representations of every hidden layer and then the final values are outputted by integrating predictions of different layers together. Therefore, the prediction performance of SAE-based models should be improved since better pretraining and fine-tuning strategies are adopted.

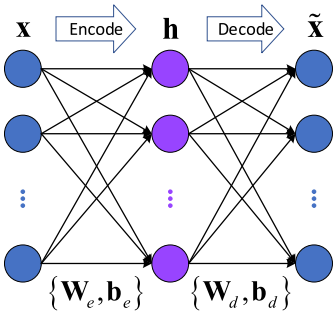

Fig. 1. Structure of autoencoder.

The layout of this article is given as follows. In Section II, the basic SAE and gated neurons-related methodologies are shortly overviewed. Then, the GSTAE model is proposed in Section III and the soft sensor modeling framework based on GSTAE is shown in Section IV. After that, the effectiveness and feasibility of the proposed approach are demonstrated by constructing two soft sensors for real industrial applications in Section V. Finally, conclusions are made in Section VI.

II. Basic Stacked Autoencoder and Gated Unit Theories

A. Autoencoder and Stacked Autoencoder

An autoencoder is a kind of DL method that extracts the hidden layer features and learns parameters in an unsupervised way by keeping its inputs and the outputs as consistent as possible (determined by information loss). The basic structure of the autoencoder is shown in Fig. 1, which contains three layers: 1) an input layer; 2) a hidden layer; and 3) an output layer.

Suppose the input data of AE are $\mathbf{x} = [x(1), x(2), \ldots, x(u)]^T \in \mathbb{R}^{u\times 1}$, where u is the dimension of input data. The input data are encoded and decoded by two nonlinear mappings, respectively, as follows:

$$\mathbf{h} = f_e(\mathbf{W}_e \bullet \mathbf{x} + \mathbf{b}_e) \tag{1}$$

$$\tilde{\mathbf{x}} = f_d(\mathbf{W}_d \bullet \mathbf{h} + \mathbf{b}_d) \tag{2}$$

where $\mathbf{h} = [h(1), h(2), \ldots, h(v)]^T \in \mathbb{R}^v$ is the encoded feature of the hidden layer, $\tilde{\mathbf{x}} = [\tilde{x}(1), \tilde{x}(2), \ldots, \tilde{x}(u)]^T \in \mathbb{R}^u$ is the output data, namely, the reconstructed input data, and $f_e(\bullet)$ and $f_d(\bullet)$ are the nonlinear activation functions that are

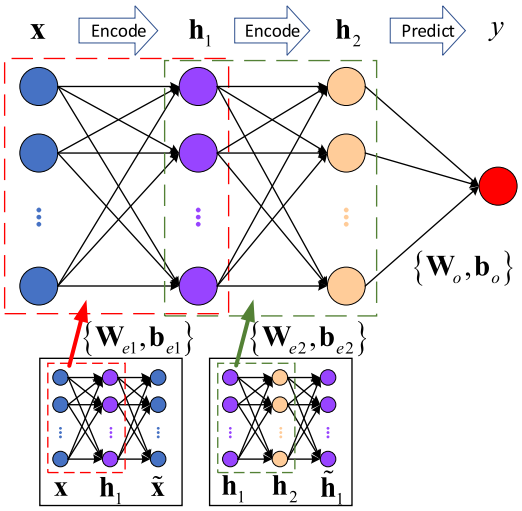

Fig. 2. Structure of SAE with two hidden layers.

used in the encode and decode procedures, such as sigmoid, hyberbolic tangent, or the rectified linear unit (ReLU) functions. The parameter set of weights and biases that the model needs to learn is $\theta_{\mathrm{AE}} = \{\mathbf{W}_e, \mathbf{b}_e; \mathbf{W}_d, \mathbf{b}_d\}$, which should make sure that the reconstructed error of input data is as small as possible. Hence, the loss function and the object function for parameter learning of AE are defined, respectively, as follows:

$$L_{\mathrm{rec}}(\theta_{\mathrm{AE}}) = \frac{1}{N}\sum_{n=1}^{N}\left\|\tilde{\mathbf{x}}_n - \mathbf{x}_n\right\|^2 \tag{3}$$

$$\theta^*_{\mathrm{AE}} = \arg\min_{\theta_{\mathrm{AE}}} L_{\mathrm{rec}}(\theta_{\mathrm{AE}}) \tag{4}$$

where N is the total number of training samples. Then, the parameters are learned by gradient descent and backpropagation algorithms.

The performance of a simple AE with a single hidden layer is customarily unsatisfactory for many complex tasks. Therefore, it is a common practice to stack several hidden layers of AE into together to increase the network depth, which is called as SAE. For a better understanding, a schematic structure of an SAE with two hidden layers is shown in Fig. 2.

The learning process of SAE is divided into two parts, namely, well-known layerwise pretraining and fine-tuning procedure. In the stage of layerwise pretraining, every two adjacent layers of SAE are trained as the encoding part of an AE just as the bottom of Fig. 2 shows. After the first AE is trained, the features $\mathbf{h}_1 = [h_1(1), h_1(2), \ldots, h_1(v_1)]^T$ (v_1 denotes the dimension of the first hidden layer) encoded from the original input data are then treated as the input of the second AE and the second hidden layer $\mathbf{h}_2$ can be trained in a similar way by minimizing the reconstruction error between $\tilde{\mathbf{h}}_1$ and $\mathbf{h}_1$. By analogy, all hidden layers of SAE can be pretrained by this kind of unsupervised learning method. After that, an extra layer will also be stacked for the output task by the mapping $\tilde{\mathbf{y}} = f_o(\mathbf{W}_o \bullet \mathbf{h}_2 + \mathbf{b}_o)$ and some labeled data samples will be used for end-to-end fine-tuning of the entire network's parameters $\theta_{\mathrm{SAE}} = \{\mathbf{W}_{e1}, \mathbf{b}_{e1}; \mathbf{W}_{e2}, \mathbf{b}_{e2}; \mathbf{W}_o, \mathbf{b}_o\}$. In this stage, the loss function for training a regression network

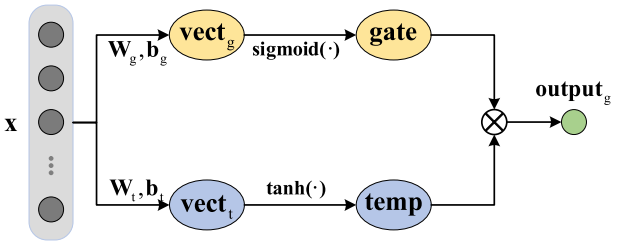

Fig. 3. Working mechanism of a gate.

is set as follows:

$$L_{f-t}(\theta_{\mathrm{SAE}}) = \frac{1}{N_L}\sum_{n=1}^{N_L}\left\|\tilde{\mathbf{y}} - \mathbf{y}\right\|^2 \tag{5}$$

where N_L denotes the total number of the labeled samples for training, $\tilde{\mathbf{y}}$ is the predicted value vector, and $\mathbf{y}$ is the target real value. Minimize the loss function $L_{f-t}(\theta_{\mathrm{SAE}})$ by using the gradient descent algorithm so that the parameters can be tuned to an appropriate numerical combination.

B. Gated Neurons

Gated neurons are mainly employed in long short-term memory unit (LSTM) and gated recurrent unit (GRU) in order to modulate the flow of information inside the unit [28]–[30]. For example, a forget gate is used to determine or control what information in the last time step can be kept in the current step. A gate is actually a kind of fully connected layer, whose input is a vector and the output is a real vector with values between 0 and 1. Mathematically, the general formula of a gate can be defined as follows:

$$\mathbf{g}(\mathbf{x}) = \sigma(\mathbf{W} \bullet \mathbf{x} + \mathbf{b}) \tag{6}$$

where σ denotes the sigmoid function: $\sigma(z) = 1/(1+e^{-z})$ and $\mathbf{W}$ and $\mathbf{b}$ are the corresponding weight matrix and bias vector, respectively. To be intuitive, the diagram of how a gate neuron makes sense is shown in Fig. 3. The key nodes of Fig. 3 can be described as follows:

$$\mathbf{vect}_g = \mathbf{W}_g \bullet \mathbf{x} + \mathbf{b}_g \tag{7}$$

$$\mathbf{gate} = \mathrm{sigmoid}(\mathbf{vevt}_g) \tag{8}$$

$$\mathbf{vect}_t = \mathbf{W}_t \bullet \mathbf{x} + \mathbf{b}_t \tag{9}$$

$$\mathbf{temp} = \tanh(\mathbf{vect}_t) \tag{10}$$

$$\mathbf{output}_g = \mathbf{temp} \otimes \mathbf{gate} \tag{11}$$

where "$\otimes$" denotes the entrywise product.

A gate value can be obtained by transmitting the input vector through nonlinear mapping and sigmoid function. In turn, the gate value is used as a switch to determine how much information mapped from the input vector will be reserved.

III. Gated Stacked Target-Related Autoencoder

To extract the abstract feature representation of input data, an unsupervised layerwise pretraining procedure is used. Hence, the features extracted by these hidden layers may contain some useless information for soft sensor modeling. Meanwhile, taking the reconstruction error as the only loss item may increase the possibility of overfitting for the network

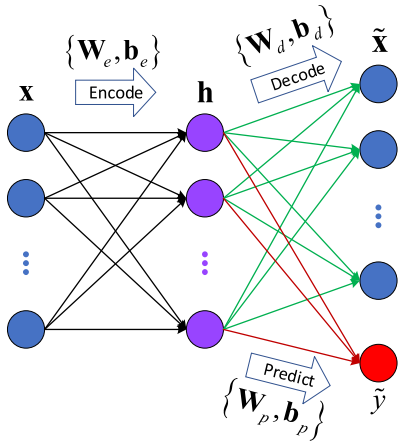

Fig. 4. Learning process of TAE and STAE.

that makes SAE always try to output a copied version of the original input data. For these reasons, the fine-tuning procedure will become more complex since an inappropriate initial state has been learned especially when facing a much deep network that contains many parameters. Besides, different hidden layers are exactly different representations of the input data, and features extracted by different layers are almost linearly independent since the higher-level hidden layer is mapped from a relatively shallow layer through nonlinear transformations. However, the traditional SAE only utilizes the features representation of the last hidden layer for the output modeling task. Therefore, there is still much room for improvement in the full use of the information contained in different hidden layers of the network.

To deal with those problems and improve the performance of SAE-based soft sensors, we propose the novel gated GSTAE in this article.

A. Target-Related Autoencoder and Stacked Target-Related Autoencoder

The goal of layerwise pretraining is to obtain the relatively good initial weights and biases of SAE, which is later used to output a target value. In other words, it is a more reasonable AE that puts more importance to extract features that are highly related for output prediction. Recently, Yuan *et al.* have proposed a VWSAE that tried to obtain hierarchical output-related feature representation layer by layer by correlation analysis with the output variable. However, only considering a linear correlation is insufficient and it is also difficult to find a perfect correlation measurement.

Therefore, we directly add the predicted error term of the target value to the loss function of the AE model when extracting features layer by layer, instead of only taking the reconstruction error into account, which is called the target-related autoencoder (TAE). In this way, it is expected that the features of every layer are not only representations of initial input data but also effective explanatory variables of the target value.

The learning process of TAE is shown in Fig. 4 from which we can see that the encoded features $\mathbf{h} = [h(1), h(2), \ldots, h(v)]^T$ are not only decoded into the reconstructed input data through the mapping $\tilde{\mathbf{x}} = f_d(\mathbf{W}_d \bullet \mathbf{h} + \mathbf{b}_d)$ but also utilized for predicting the target value through the mapping $\tilde{\mathbf{y}} = f_p(\mathbf{W}_p \bullet \mathbf{h} + b_p)$. The training loss function is accordingly defined as follows:

$$L_{\text{TAE}} = \frac{1}{N}\sum_{n=1}^{N}\left\|\tilde{\mathbf{x}}(n) - \mathbf{x}(n)\right\|^2 + \frac{\lambda}{N}\sum_{n=1}^{N}\left\|\tilde{\mathbf{y}}(n) - \mathbf{y}(n)\right\|^2 \tag{12}$$

where λ is a tradeoff coefficient to balance the weights of reconstruction loss and prediction loss in the loss function. The parameters needed to learn is $\theta_{\text{TAE}} = \{\mathbf{W}_e, \mathbf{b}_e; \mathbf{W}_d, \mathbf{b}_d; \mathbf{W}_p, \mathbf{b}_p\}$ that should be obtained by minimizing (6). In this way, TAE is a supervised model that needs the labeled data for training.

Once the hidden layer has been well trained, it can be further used for the direct output task or stacking into the deep neural network called stacked TAE (STAE), which is shown on the left side of the black dotted line in Fig. 5. The lower-level features are used as the input of the next TAE, and the loss function of the kth ($k \geq 2$) TAE can be similarly defined as follows:

$$L_{k-TAE} = \frac{1}{N}\sum_{n=1}^{N}\left\|\tilde{\mathbf{h}}_{k-1}(n) - \mathbf{h}_{k-1}(n)\right\|^2 + \frac{\lambda}{N}\sum_{n=1}^{N}\left\|\tilde{\mathbf{y}}(n) - \mathbf{y}(n)\right\|^2 \tag{13}$$

where $\mathbf{h}_{k-1}(n)$ and $\tilde{\mathbf{h}}_{k-1}(n)$ are the input and reconstructed input data of the kth ($k \geq 2$) TAE.

B. Gated Staked Target-Related Autoencoder

The target-related information is considered in the learning process of STAE, which means that the extracted features of every hidden layer have more output-relevant characteristics and the correlation strength will be cumulatively enhanced through the layerwise learning process. In other words, each layer is a different level of representation for connecting inputs and outputs, and the shallower the layer, the more input information it retains and the weaker correlation with the output it has. In order to improve model performance, we concentrate on taking full advantage of different levels' feature representations. Although it is recognized that the higher hidden layer tends to extract more abstract features, the features of shallow layers still have their modeling value. To this end, the GSTAE is proposed here as shown in Fig. 5, which introduces the gated neuron to further mine and integrate information in different layers.

Suppose the depth of STAE is n layers, and $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n$ denote the features from the first layer to the last layer, respectively. For the kth layer, the features are first used to generate a gated value as follows:

$$g_k = \sigma\left(\mathbf{W}_{g_k} \bullet \mathbf{h}_k + b_{g_k}\right) \tag{14}$$

where $\mathbf{W}_{g_k} \in \mathbb{R}^{1\times n_k}$, $\mathbf{h}_k \in \mathbb{R}^{n_k\times 1}$, and $b_{g_k} \in \mathbb{R}^{1\times 1}$ and n_k is the number of neurons in the kth hidden layer. Meanwhile,

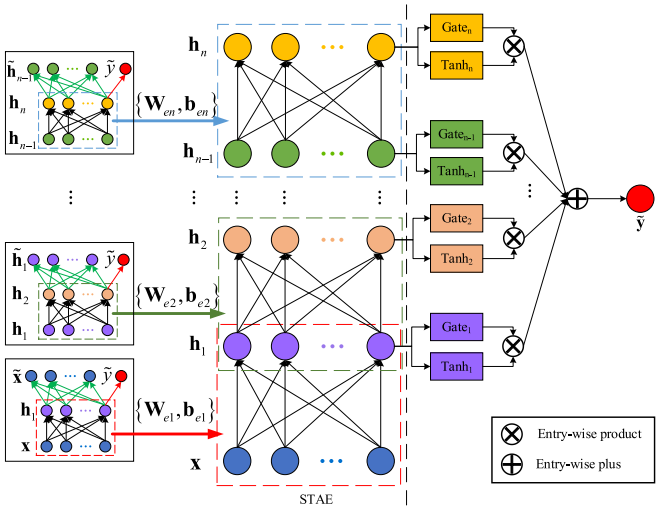

Fig. 5. Learning process of GSTAE.

the features $\mathbf{h}_k$ are utilized to generate an alternative output y_k according to the following mapping:

$$y_k = \tanh\left(\mathbf{W}_{o_k} \bullet \mathbf{h}_k + b_{o_k}\right) \tag{15}$$

and then the final output can be obtained by integrating all alternative outputs as follows:

$$\begin{aligned}\tilde{y} &= \sum_{k=1}^{n} g_k \circ y_k \\ &= \sum_{k=1}^{n} \sigma\left(\mathbf{W}_{g_k} \bullet \mathbf{h}_k + b_{g_k}\right) \circ \tanh\left(\mathbf{W}_{o_k} \bullet \mathbf{h}_k + b_{o_k}\right)\end{aligned} \tag{16}$$

where symbol "$\circ$" represents the entrywise product. In this way, gated neurons are used to determine what weight each layer accounts for the final output.

After that, a backward fine-tuning procedure is conducted to solve the parameter set $\theta_g = \{\mathbf{W}_{e_1}, \mathbf{b}_{e_1}; \cdots; \mathbf{W}_{e_n}, \mathbf{b}_{e_n}; \mathbf{W}_{g_1}, b_{g_1}; \cdots; \mathbf{W}_{g_n}, b_{g_n}; \mathbf{W}_{o_1}, b_{o_1}; \cdots; \mathbf{W}_{o_n}, b_{o_n}\}$ through minimizing the following unconstrained optimization problem:

$$\min_{\theta_g} \Gamma_{\mathrm{GSTAE}} = \frac{1}{N}\sum_{n=1}^{N}\left\|\tilde{\mathbf{y}}(n) - \mathbf{y}(n)\right\|^2 \tag{17}$$

which is equivalent to minimizing the mean-square error (MSE) between predicted values $\tilde{\mathbf{y}}$ and real values $\mathbf{y}$.

The detailed training algorithm to train a GSTAE network is summarized as Table I.

TABLE I
ALGORITHM 1: TRAINING ALGORITHM OF GSTAE

Input: $\mathbf{x}, \mathbf{y}$: original datasets; λ : trade-off parameter of target-related term, n : number of hidden layers.

Output: $\theta_g = \{\mathbf{W}_{e_1}, \mathbf{b}_{e_1}; \cdots; \mathbf{W}_{e_n}, \mathbf{b}_{e_n}; \mathbf{W}_{g_1}, b_{g_1}; \cdots; \mathbf{W}_{g_n}, b_{g_n}; \mathbf{W}_{o_1}, b_{o_1}; \cdots; \mathbf{W}_{o_n}, b_{o_n}\}$: weights and biases of the encoding part of STAE and gated neuron.

Start:

1、Standardize the process variables set $\mathbf{x}$ into $\mathbf{x}_s$;

2、Determine the structure parameters of GSTAE;

3、**for** $k \leftarrow 1$ *to* n :

if $k = 1$:

(1) Use the standardized set $\mathbf{x}_s$ as the input data and $\mathbf{y}$ as the target values of the first TAE, and minimize equation (7) to obtain the encoding parameters $\{\mathbf{W}_{e1}, \mathbf{b}_{e1}\}$;

(2) Encode $\mathbf{x}_s$ to the first hidden layer to extract features $\mathbf{h}_1$ using $\{\mathbf{W}_{e1}, \mathbf{b}_{e1}\}$;

else:

(1) Use the features $\mathbf{h}_{k-1}$ as the input data and $\mathbf{y}$ as the target values of the k th TAE, and minimize equation (8) to obtain the encoding parameters $\{\mathbf{W}_{ek}, \mathbf{b}_{ek}\}$;

(2) Encode $\mathbf{h}_{k-1}$ to the second hidden layer to extract features $\mathbf{h}_k$;

end if

end for

4、The forward encoding parameters are used to extract layers of feature: $\mathbf{h}_k\,(k = 1, 2, \cdots, n)$, pass them into the corresponding gates and hyberbolic tangent functions to predict the final outputs. Obtain optimal parameters set $\theta_g = \{\mathbf{W}_{e_1}, \mathbf{b}_{e_1}; \cdots; \mathbf{W}_{e_n}, \mathbf{b}_{e_n}; \mathbf{W}_{g_1}, b_{g_1}; \cdots; \mathbf{W}_{g_n}, b_{g_n}; \mathbf{W}_{o_1}, b_{o_1}; \cdots; \mathbf{W}_{o_n}, b_{o_n}\}$ by solving equation (12);

5、Save structure and learning parameters of GSTAE for later online quality prediction.

End

IV. GSTAE-BASED SOFT SENSOR MODELING

The proposed GSTAE algorithm is capable to extract target-related features by layerwise encoding, and different level's features are further mined and combined together by gated neurons. The basic framework for GSTAE-based soft sensor modeling will be introduced in this section. There are mainly two steps to apply the proposed DL algorithm for quality prediction of the actual industrial process, namely, offline training and online prediction. In the procedure of offline training, the collected dataset is preprocessed and then randomly split into training set and validation set according to appropriate proportions, in which the former is the dataset used for model training and the latter for performance testing. If the model achieves relatively satisfactory and similar performance in both the training set and the validation set (if the performance on the validation set is much worse than on the training set, it means that the model is overfitting), then the parameters of the network could be saved and used for online quality forecasting. As for the to-be-predicted samples, first, they will be preprocessed in the same way as the training and validation sets. Afterward, they are fed into the well-trained GSTAE model and encoded through layer-by-layer propagation in STAE. Subsequently, the gated neurons fully extract and merge the information in different hidden layers to output predicted values. Intuitively, the framework of soft sensor modeling based on the GSTAE algorithm is displayed in Fig. 6.

For practical application, some indices will be used to evaluate the model performance quantitatively. There are mainly two indices: 1) root MSE (RMSE) and 2) R^2 (called the

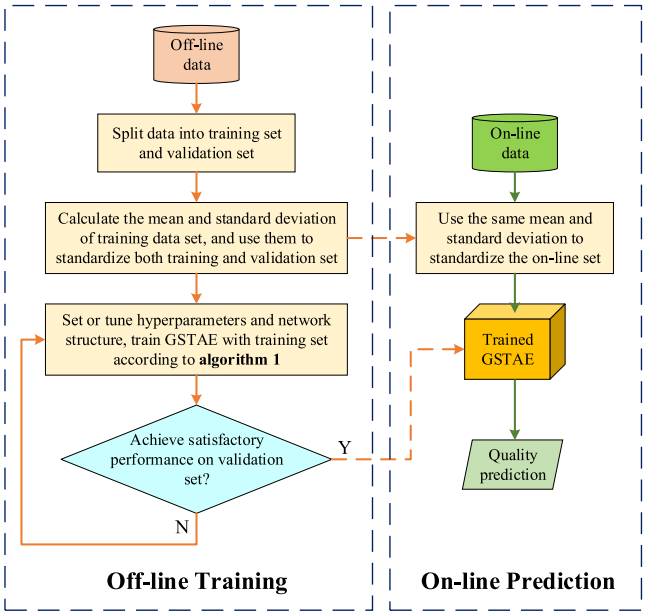

Fig. 6. Framework of GSTAE-based soft sensor modeling.

goodness of fit or coefficient of determination). RMSE focuses on the prediction errors of the entire validation or testing set which is calculated according to the following definition:

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{n=1}^{N} \left\| \mathbf{y}(n) - \tilde{\mathbf{y}}(n) \right\|^2}{N}} \tag{18}$$

where N denotes the total number of samples used for performance evaluation, $\mathbf{y}(n)$ is the true value, and $\tilde{\mathbf{y}}(n)$ is the predicted value. The smaller the RMSE, the more accurate the prediction. As another widely used index for prediction problem, R^2 is defined as

$$R^2 = 1 - \frac{\sum_{n=1}^{N} \left(\mathbf{y}(n) - \tilde{\mathbf{y}}(n) \right)^2}{\sum_{n=1}^{N} (\mathbf{y}(n) - \bar{\mathbf{y}})^2} \tag{19}$$

where $\bar{\mathbf{y}}$ denotes the mean true values of validation or testing set. R^2 represents a squared correlation between true output and predicted output, and it is usually seen as a numerical indication about the ability of the model to interpret the variance of output data. The closer R^2 is to 1, the better the prediction performance of model. According to these two indices, the effect of offline training and the performance of online prediction can be quantificationally measured.

V. Case Studies

In this section, the performance of the proposed GSTAE algorithm is validated on two real chemical industrial process cases. In the first case, we aim at demonstrating the effectiveness of STAE, and in the second case, the superiority of the GSTAE model will be validated in detail.

A. Debutanizer Column

As a benchmark case to validate data-driven methods in the soft sensor modeling domain, the debutanizer column is a vital part of the refinery process that is used for desulfuration and naphtha split by removing propane and butane from the naphtha stream [31]. The flowchart of the debutanizer column is shown in Fig. 7. To improve the control quality, several sensors are installed to monitor the process and the corresponding seven variables measured by these sensors are listed and introduced shortly in Table II. Since the main goal of the debutanizer column is to minimize the content of butane, it is of much importance to measure the content of butane in real time. However, the butane concentration is measured on the overheads of the sequential deisopentanizer column by the gas chromatograph instead of being measured on the bottom flow directly, which usually causes a large measuring delay. Besides, there are complicated nonlinearities between those variables [6], [30]. Therefore, it is full of significance to build a nonlinear soft sensor for online prediction of the content of butane.

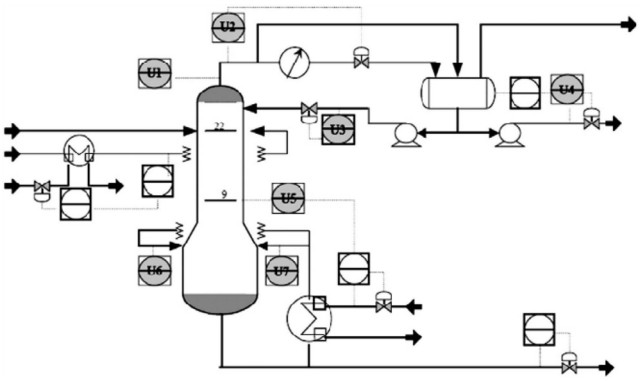

Fig. 7. Flowchart of the debutanizer column [31].

TABLE II
Process Variables for Soft Sensor on the Debutanizer Column

Input variables	Variable description
u_1	Top temperature
u_2	Top pressure
u_3	Reflux flow
u_4	Flow to next process
u_5	6-th tray temperature
u_6	Bottom temperature
u_7	Bottom pressure

In this case, the SAE model is built as the baseline method. Besides, to verify the advantage of the proposed STAE approach, a VWSAE-based [25] soft sensor is also constructed, since both of them aim to improve the pretraining effectiveness of SAE by reinforcing the supervisory function of target variables. An opensource dataset with 2394 samples is available for constructing soft sensors. According to [31], a great time-delay issue exists in the process, and it can be overcome by introducing several well-designed new variables into the original seven variables. Specifically, a total of 13 variables at the kth sampling time that are listed as

$$\begin{bmatrix} u_1(k), u_2(k), u_3(k), u_4(k), u_5(k), u_5(k-1), \\ u_5(k-2), u_5(k-3), (u_1(k)+u_2(k))/2, \\ y(k-1), y(k-2), y(k-3), y(k-4) \end{bmatrix}^T. \tag{20}$$

According to this procedure, first 1004 samples in the original dataset are used to obtain 1000 new samples as the training

TABLE III
NETWORK STRUCTURE OF SAE, VWSAE, AND STAE

Layer	Input	Hidden 1	Hidden 2	Hidden 3	Output
Number of neurons	13	10	7	4	1

set, and the next 304 original samples are utilized to obtain 300 new samples as the validation set. Afterward, the following 504 original samples are selected to build the testing set.

It is a key point to determine the hyperparameters of the networks. To the best of our knowledge, the common hyperparameter search methods include the trial-and-error method, grid search method, random search method, and the Bayesian optimization method. Since the data size and network complexity are not very large, it is unnecessary and inefficient to use the grid search and the Bayesian optimization. Therefore, we combine the random search and trial-and-error method to determine the optimal hyperparameters. First, generate several groups of hyperparameters and run several times to obtain experimental results. After that, compare the results at different values of every hyperparameter and find the roughly optimal intervals. Furthermore, use the trial-and-error method to search the detailed optimal hyperparameter set. Besides, the structural parameters of networks are determined in a simpler way. As for the number of hidden layers, it is determined mainly by the complexity of the data being processed. In this case, we found that possessing three hidden layers is the best choice since less layers will lead to poor performance and more layers will cause more computational burdens without distinct performance improvement. While the search spaces of the neuron number in every hidden layer are actually more continuous, so imperceptible changes will not cause distinct influence. In addition, on different devices, the optimal parameter set may be slightly different. The detailed structural parameters are listed in Table III, and the common network structure of three SAE-based models are set consistently.

The tradeoff coefficient λ is a key parameter that will influence the performance of the STAE model, so it is carefully determined and finally set as 0.5 here.

In order to validate the improvement of STAE on layerwise feature extraction, the training cost [refer to (5) for definition which is consistent for three models] curves in the fine-tuning phase are displayed in Fig. 8 first. First, the start points of the fine-tuning procedure in Fig. 8 are compared with each other, which are also the end states of pretraining. It can be seen that STAE achieves the lowest cost than two other methods before the fine-tuning procedure begins (Cost of SAE : VWSAE : STAE $\approx$ 0.89 : 0.29 : 0.17), which verifies that STAE can extract better features for the soft sensing task since the target-related information is added into its loss function in a more proper way. Besides, after 50 epochs' fine-tuning procedure, STAE got the lowest final training cost than SAE. In other words, STAE has reached a better initial point after being pretrained layer by layer so that it can find a better local optimal solution of parameters after the entire training process. After that, the soft sensor models can be used for the content prediction of butane in this case.

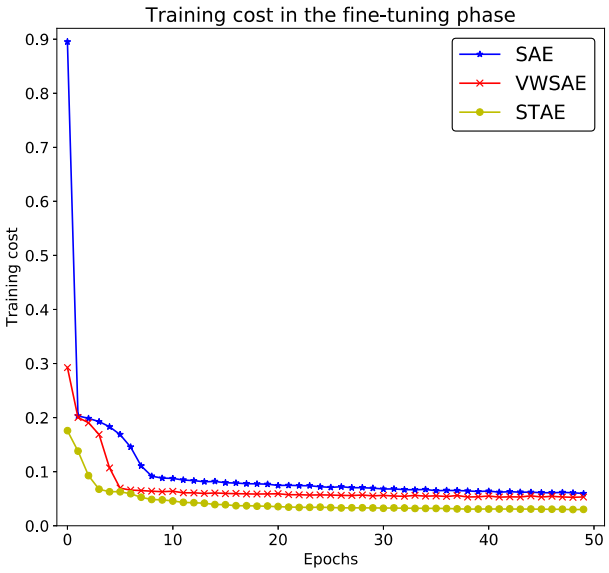

Fig. 8. Training cost curves in the fine-tuning phase for three models.

TABLE IV
EVALUATION INDICES OF THREE METHODS

Index \ Model	SAE	VWSAE	STAE
RMSE	0.047436	0.037469	0.019137
R^2	0.940481	0.962467	0.990528

To prevent evaluating model performance based on some extreme results, the average values of RMSE and R^2 after 20 times of training and predictions are calculated and listed in Table IV, which directly verifies the superiority of STAE.

Furthermore, we pick one out of 20 results of every model whose RMSE index is the closest to the average values ($\text{RMSE}_{\text{SAE}} = 0.047655$, $\text{RMSE}_{\text{VWSAE}} = 0.037652$, $\text{RMSE}_{\text{GSTAE}} = 0.019081$), and the prediction results and errors are displayed in Fig. 9. It can be found that the proposed STAE obtains the best performance since its predictive error interval is the narrowest and most symmetric about 0 (from -0.04730 to $+0.07188$). However, the biggest difference among these prediction curves of three methods is the prediction accuracy of high-value samples, which can be seen more clearly in Fig. 10 in which the horizontal axis is the real value and the vertical axis is the predicted value. Therefore, the closer the scattered point is to the main diagonal (yellow dashed line), the better the prediction performance. In the interval of [0.0, 0.6], the performance of STAE is slightly better than SAE and VWSAE. While in the big value interval of [0.6, 0.9], the STAE shows obvious superiority compared to the other methods since it predicts the content of butane more accurately which is emphasized with a purple oval in Fig. 10.

To explore what leads to the error of big-value prediction, the distribution histogram of the target values is presented in Fig. 11, in which the black solid line is a normal distribution estimation (NDE) with the mean and stand deviation of

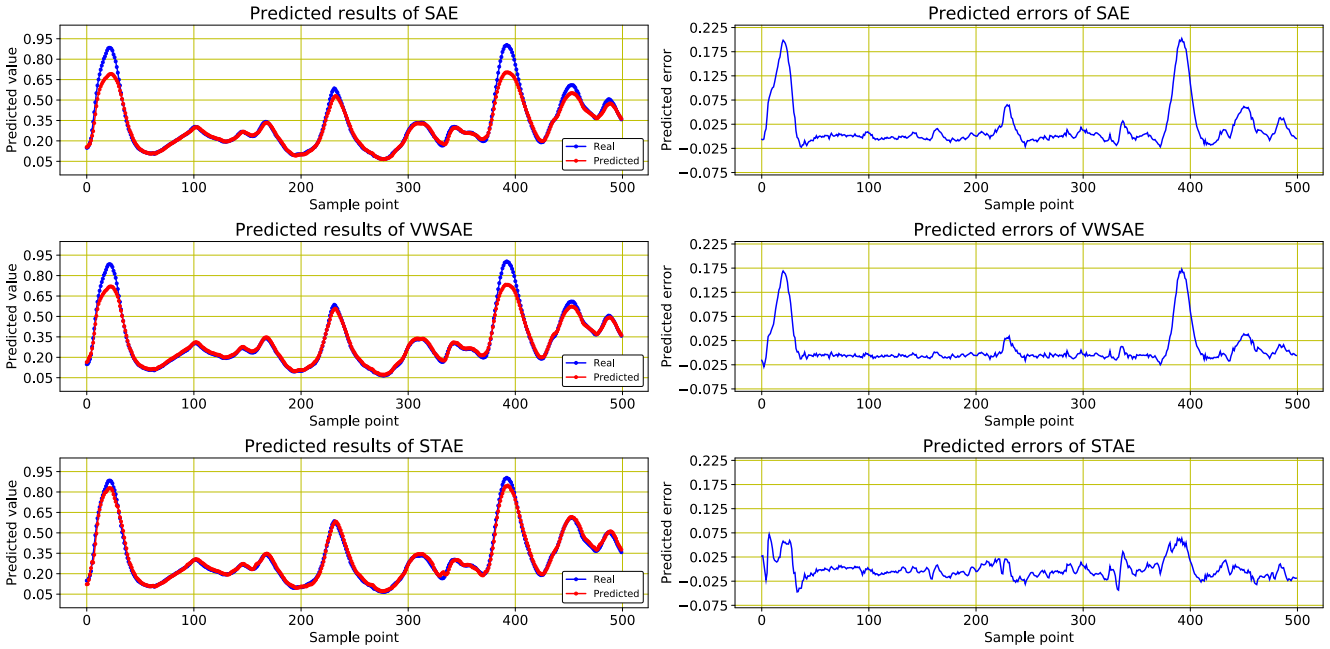

Fig. 9. Prediction results and errors of three methods.

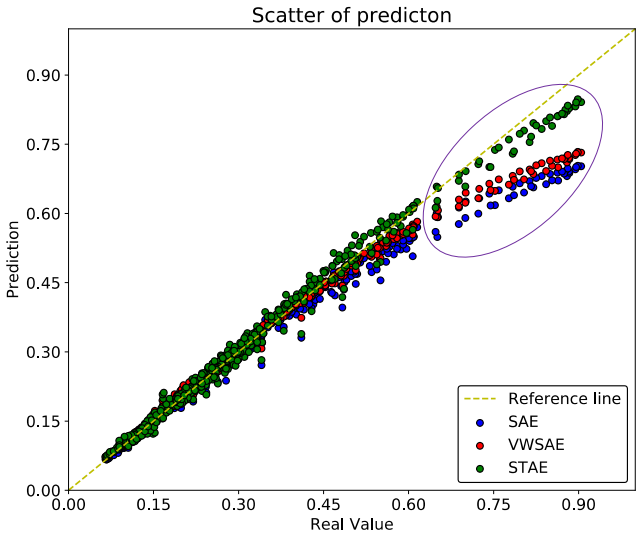

Fig. 10. Scatter diagram of prediction results of three methods.

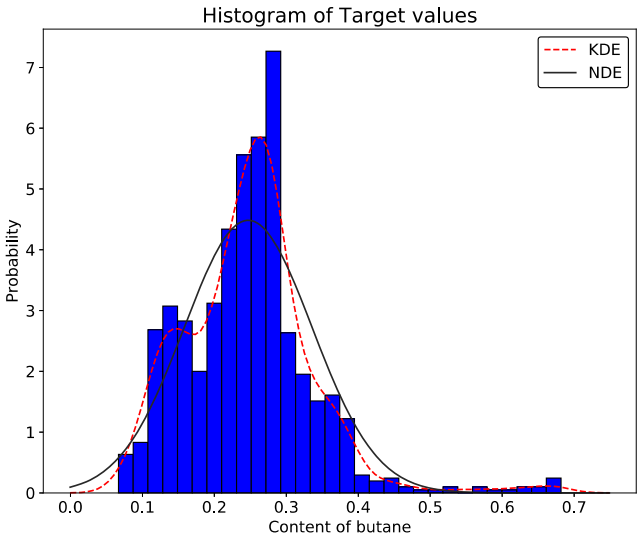

Fig. 11. Distribution histogram of target values.

the content of butane in the training set while the red dashed line is kernel density estimation (KDE). It can be seen from Fig. 11 that the target values do not follow a normal distribution, and it shows an asymmetry since there are still a certain proportion of samples in the interval of [0.5, 0.7] which are away from the mean value. Therefore, the prediction difficulty of large-value samples in the testing set is increased, especially when the values are bigger than 0.7 since they are almost absent from the training set. Unlike VWSAE, which only reinforces the guidance of target values according to the linear correlation coefficient during pretraining, STAE directly takes the predictive loss into consideration so as to introduce the guidance of target values in an intelligent way. Therefore, STAE pays more attention to the samples which lead to big prediction error, rather than just samples that make up a large percentage of the distribution. From the perspective of this industrial process, it is more than significant to predict the butane content as accurately as possible when its values are relatively large because this would help to detect abnormal process status in time which leads to those excessive values.

In short, taking target-related information into account during a layerwise pretraining procedure can enhance the sensitivity to the to-be-predicted value of SAE so that it can extract more useful features for improving the performance of quality prediction.

B. CO_2 Absorption Column

The CO_2 absorption column is a vital processing equipment in the ammonia synthesis process, which provides the primary material NH_3 for the subsequent urea synthesis process.

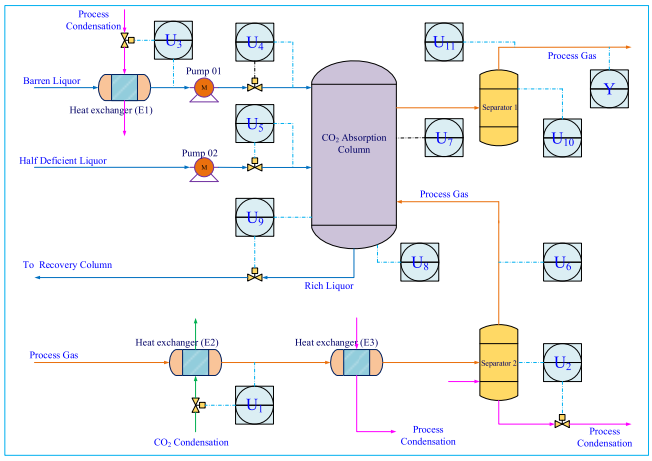

Fig. 12. Flowchart of CO_2 absorption column.

TABLE V
DESCRIPTION OF THE PROCESS VARIABLES IN CO_2 ABSORPTION COLUMN

Tags	Descriptions
U1	Pressure of process gas into E3
U2	Liquid level of Separator 2
U3	Temperature of barren liquor at E1 exit
U4	Flow rate of barren liquor to CO_2 Absorption Column
U5	Flow rate of half deficient liquor to CO_2 Absorption Column
U6	Temperature of process gas at exit of Separator 2
U7	Differential pressure of process gas at entrance of CO_2 Absorption Column
U8	Temperature of rich liquor at exit of CO_2 Absorption Column
U9	Liquid level of CO_2 Absorption Column
U10	High level alarming of Separator 1
U11	Pressure of process gas at the exit
Y	Concentration of residual CO_2 in process gas

Gaseous hydrogen produced by the methane decarburization unit is used as a raw material in the process of ammonia synthesis. But there are still some carbon elements in the process gas in the form of gaseous CO_2, which need to be segregated to obtain pure materials and then be further exploited in the urea synthesis process. Therefore, the concentration of residual CO_2 is a critical index which can influence the production quality of the final ammonia synthesis unit and should be real-time monitored. However, the measurement of the CO_2 concentration is anything but easy and cheap. As the flowchart shown in Fig. 12, an expensive mass spectrometer needs to be installed on the exit pipe of 06F001. Therefore, a soft sensor should be built in order to help to monitor the concentration of CO_2 and reduce the measurement cost. There are totally 11 process variables and one quality variable here which are listed and shortly described in Table V.

In this case, we collect 2000 samples from the database of DCS. First, 1500 samples are utilized as a training set and the next 500 samples are used as the testing set. For comparison, we construct four soft sensor models, including SAE, VWSAE, STAE, and GSTAE. The determination of hyperparameters is the same as described in the last case. Structurally, the basic network structure is set as consistent that all these four models have three hidden layers and one output layer with 9, 7, 4, and 1 neurons, respectively. The $L2$ regularization term is added into the loss function of each model to avoid overfitting which is defined as follows:

$$R_{L2} = \alpha \|\mathbf{W}\|^2 \tag{21}$$

where the penalty coefficient $\alpha = 0.005$ and $\mathbf{W}$ denotes all the weights in the network. Besides, λ here is set as 0.5.

In this case, to further show the effectiveness of the proposed GSTAE method, we introduced more numerical metrics. The mean absolute error (MAE) and mean absolute percentage error (MAPE) are defined as follows:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|\tilde{\mathbf{y}}(n) - \mathbf{y}(n)\right| \tag{22}$$

$$\text{MAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{\tilde{\mathbf{y}}(n) - \mathbf{y}(n)}{\mathbf{y}(n)}\right|. \tag{23}$$

TABLE VI
EVALUATION INDICES OF FOUR METHODS

Index \ Model	SAE	VWSAE	STAE	GSTAE
RMSE	0.003731	0.003393	0.002927	0.002434
R^2	0.8625	0.8863	0.9154	0.9415
MAE	0.002947	0.002735	0.002360	0.001923
MAPE	0.9837	0.9162	0.7933	0.6511

The average evaluation indices after 20 times of experiments on the testing set are listed in Table VI. Compared with the other three models, GSTAE achieves the best performance according to all four indices.

To directly compare the performance of these models, the test results that are closest to the averages of the corresponding evaluation indicators are displayed in Fig. 13. According to the metrics in Table VI and the results shown in Fig. 13, STAE performs better than VWSAE, especially the high values, which is consistent with the conclusion of the last case. Besides, it can be seen from Fig. 13 that the red line (predicted results) and the blue line (real values) have the highest coincidence in the lower right subgraph. In other words, GSTAE achieves the best predictive performance for target outputs, which shows a significant improvement in SAE-based soft sensing application. Although VWSAE and STAE use the target values to realize better feature extraction during the layerwise pretraining process, they do not take full advantage of those well-learned representations. Instead, only the features in the last layer are directly connected with the output layer. While for GSTAE, connections are established between each hidden layer and the output layer through gated neurons. Therefore, a more precise model can be obtained since the model uses more feature representations, which are at different abstraction levels, to predict the target output. To further testify the effectiveness of the proposed GSTAE model, the rectangular box plot of prediction errors is displayed in Fig. 14, which shows that the prediction errors of GSTAE are more densely distributed around 0. In other words, GSTAE can make a more

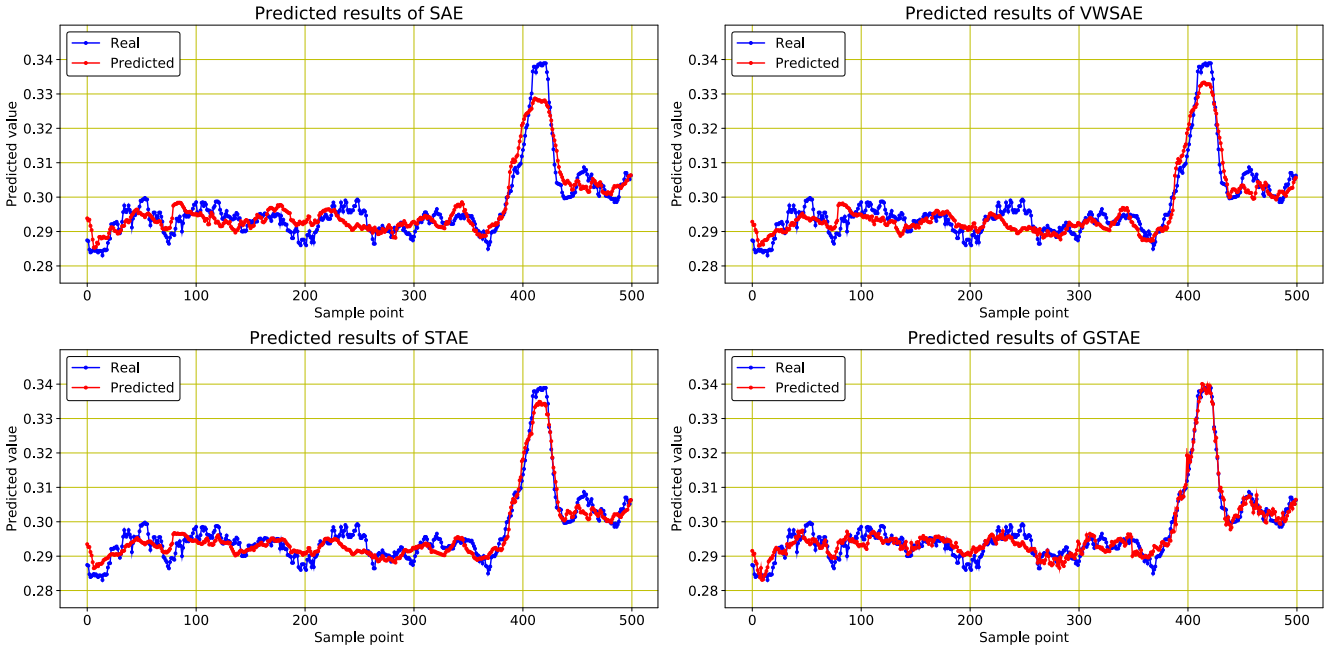

Fig. 13. Prediction performance on the testing dataset.

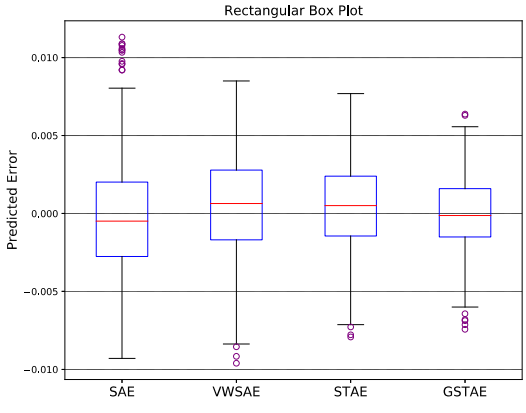

Fig. 14. Rectangular box plot of prediction error.

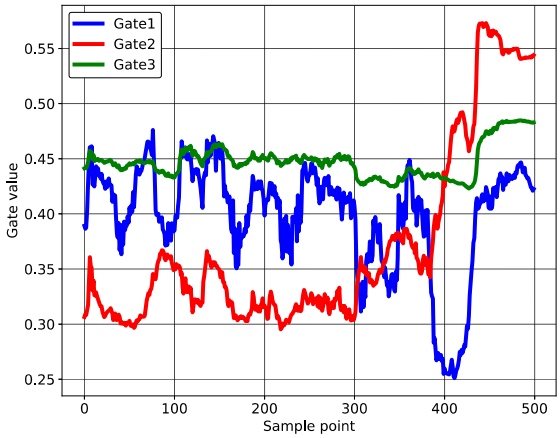

Fig. 15. Gate values of three gates on the testing set.

TABLE VII
CONFIDENCE INTERVALS OF PREDICTION ERROR ($\times 10^{-2}$)

Parameter \ Model	SAE	VWSAE	STAE	GSTAE
μ	-0.0231	0.0544	0.0463	-0.0026
σ	0.3724	0.3349	0.2890	0.2434
$[\mu-3\sigma, \mu+3\sigma]$	[-1.1402, 1.0941]	[-0.9503, 1.0592]	[-0.8205, 0.9133]	[-0.7328, 0.7275]

stable and accurate prediction. Furthermore, the normal distribution parameters and $\mu \pm 3\sigma$ intervals of prediction errors are computed and listed in Table VII, in which the GSTAE holds a smaller standard deviation and a smaller absolute mean error than others so that its interval is the narrowest and is the most symmetric about 0. Therefore, GSTAE performs better and is more stable for the prediction application.

Now that the gated neurons are utilized to control the information flow from each hidden layer to target output, the gate values need to be investigated here which are shown in Fig. 15. The blue, red, and green lines represent the values of gates 1, 2, and 3 on the testing set, respectively. By introducing gated neurons, the weights and the contributions of different hidden layers are quantified. Their corresponding average values are 0.3709, 0.3984, and 0.4476. In other words, they account for 30.5%, 32.7%, and 36.8% of the final output. Based on the numerical index and curves shown in Fig. 15, we can conclude that the higher-level features do contribute more to the output, but the features of lower layers are also responsible for a certain percentage of it. For different samples such as samples 380–430, the ratio of different gate's values varies, which helps to improve the predictive accuracy of every sample. In short, utilizing the gated neurons is verified to be effective and feasible.

VI. Conclusion

In this article, taking the shortcomings of conventional SAE into consideration, a DL model called GSTAE was proposed to improve the performance of the SAE-based model for the soft sensing application. To begin with, the proposed STAE introduced the predicted loss item of the target value into the original loss function of a common autoencoder when conducts a layerwise pretraining process so that the target-related information is reasonably considered. After the pretraining procedure of STAE, every hidden layer has its specific feature representation which not only contains the information of input but also can be used for predicting target values. Therefore, gated neurons were utilized to extract and control the information flow from different hidden layers to the output layer, and the entire model is called GSTAE. By this way, the contributions of different layers to output values can be quantified. Moreover, the effectiveness and superiority of the proposed approaches were verified by comparing with other SAE-based methods in two real industrial processes: 1) debutanizer column and 2) CO_2 absorption column. The proposed method should be helpful for improving the performance of soft sensors constructed from SAE-based methods.

However, there are some limitations of the proposed approach that need to be paid attention to. First, GSTAE was quite sensitive to samples with the poor-quality label since the supervision from target values are enhanced both in pretraining and fine-tuning process. Second, the computational burden will increase linearly with the depth of the network because every hidden layer was followed by a gated neuron in the fine-tuning process. In this case, timely removal of low-value gate connections to reduce unnecessary computational load is of great importance.

References

[1] Y. Liu, C. Yang, K. L. B. Chen, and Y. Yao, "Domain adaptation transfer learning soft sensor for product quality prediction," *Chemometr. Intell. Lab. Syst.*, vol. 192, Sep. 2019, Art. no. 103813.

[2] S. Khatibisepehr, B. Huang, and S. Khare, "Design of inferential sensors in the process industry: A review of Bayesian methods," *J. Process. Control*, vol. 23, no. 11, pp. 1575–1596, 2013.

[3] W. Zheng, Y. Liu, Z. Gao, and J. Yang, "Just-in-time semi-supervised soft sensor for quality prediction in industrial rubber mixers," *Chemometr. Intell. Lab. Syst.*, vol. 180, pp. 36–41, Sep. 2018.

[4] W. Yan, D. Tang, and Y. Lin, "A data-driven soft sensor modeling method based on deep learning and its application," *IEEE Trans. Ind. Electron.*, vol. 64, no. 5, pp. 4237–4245, May 2017.

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

[6] L. Yao and Z. Ge, "Big data quality prediction in the process industry: A distributed parallel modeling framework," *J. Process. Control*, vol. 68, pp. 1–13, Aug. 2018.

[7] V. Gopakumar, S. Tiwari, and I. Rahman, "A deep learning based data driven soft sensor for bioprocesses," *Biochem. Eng. J.*, vol. 136, pp. 28–39, Aug. 2018.

[8] Z. Ge, Z. Song, S. Ding, and B. Huang, "Data mining and analytics in the process industry: The role of machine learning," *IEEE Access*, vol. 5, pp. 20590–20616, 2017.

[9] A. Majumder, L. Behera, and V. K. Subramanian, "Automatic facial expression recognition system using deep network-based data fusion," *IEEE Trans. Cybern.*, vol. 48, no. 1, pp. 103–114, Jan. 2018.

[10] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran, 2012.

[11] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur, "Recurrent neural network based language model," in *Proc. 11th Annu. Conf. Int. Speech Commun. Assoc. (Interspeech)*, vol. 2, 2010, pp. 1045–1048.

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

[13] B. Du, W. Xiong, J. Wu, L. Zhang, L. Zhang, and D. Tao, "Stacked convolutional denoising auto-encoders for feature representation," *IEEE Trans. Cybern.*, vol. 47, no. 4, pp. 1017–1027, Apr. 2017, doi: 10.1109/TCYB.2016.2536638.

[14] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[15] H. Zhou, G. Huang, Z. Lin, H. Wang, and Y. C. Soh, "Stacked extreme learning machines," *IEEE Trans. Cybern.*, vol. 45, no. 9, pp. 2013–2025, Sep. 2015, doi: 10.1109/TCYB.2014.2363492.

[16] G. Hinton, S. Osindero, and Y-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.

[17] W. Shao, Z. Ge, and Z. Song, "Semisupervised Bayesian Gaussian mixture models for non-Gaussian soft sensor," *IEEE Trans. Cybern.*, early access, Nov. 08, 2019, doi: 10.1109/TCYB.2019.2947622.

[18] Y. Wang, Z. Pan, X. Yuan, C. Yang, and W. Gui, "A novel deep learning based fault diagnosis approach for chemical process with extended deep belief network," *ISA Trans.*, vol. 96, pp. 457–467, Jan. 2020, doi: 10.1016/j.isatra.2019.07.001.

[19] W. Shao, Z. Ge, Z. Song, and K. Wang, "Nonlinear industrial soft sensor development based on semi-supervised probabilistic mixture of extreme learning machines," *Control Eng. Pract.*, vol. 91, Oct. 2019, Art. no. 104098, doi: 10.1016/j.conengprac.2019.07.016.

[20] S. Kamat and K. Madhavan, "Developing ANN based virtual/soft sensors for industrial problems," *IFAC-PapersOnLine*, vol. 49, no. 1, pp. 100–105, 2016.

[21] C. Shang, F. Yang, D. Huang, and W. Lyu, "Data-driven soft sensor development based on deep learning technique," *J. Process. Control*, vol. 24, no. 3, pp. 223–233, 2014.

[22] L. Yao and Z. Ge, "Deep learning of semisupervised process data with hierarchical extreme learning machine and soft sensor application," *IEEE Trans. Ind. Electron.*, vol. 65, no. 2, pp. 1490–1498, Feb. 2018.

[23] A. Makhzani and B. Frey, "*k*-sparse autoencoders," 2014. [Online]. Available: https://arxiv.org/abs/1312.5663.

[24] A. Rasmus, M. Berglund, M. Honkala, M. Berglund, and T. Raiko, "Semi-supervised learning with ladder networks," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran, 2015, pp. 3546–3554.

[25] X. Yuan, B. Huang, Y. Wang, C. Yang, and W. Gui, "Deep learning-based feature representation and its application for soft sensor modeling with variable-wise weighted SAE," *IEEE Trans. Ind. Inf.*, vol. 14, no. 7, pp. 3225–3243, Jul. 2018.

[26] X. Luo, X. Li, Z. Wang, and J. Liang, "Discriminant autoencoder for feature extraction in fault diagnosis," *Chemometr. Intell. Lab. Syst.*, vol. 192, Sep. 2019, Art. no. 103814.

[27] B. Shen, L. Yao, and Z. Ge, "Nonlinear probabilistic latent variable regression models for soft sensor application: From shallow to deep structure," *Control Eng. Pract.*, vol. 94, Jan. 2020, Art. no. 104198, doi: 10.1016/j.conengprac.2019.104198.

[28] H. Sak, A. W. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Proc. 15th Annu. Conf. Int. Speech Commun. Assoc. (Interspeech)*, 2014, pp. 338–342.

[29] J. Chung, C. Gulcehre, K. H. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," Dec. 2014. [Online]. Available: arXiv:1412.3555.

[30] X. Yuan, L. Li, and Y. Wang, "Nonlinear dynamic soft sensor modeling with supervised long short-term memory network," *IEEE Trans. Ind. Inf.*, vol. 16, no. 5, pp. 3168–3176, May 2020, doi: 10.1109/TII.2019.2902129.

[31] L. Fortuna, S. Graziani, and M. G. Xibilia, "Soft sensors for product quality monitoring in debutanizer distillation columns," *Control Eng. Pract.*, vol. 13, no. 4, pp. 499–508, 2005.

[32] X. Yuan, Z. Ge, B. Huang, Z. Song, and Y. Wang, "Semisupervised JITL framework for nonlinear industrial soft sensing based on locally semisupervised weighted PCR," *IEEE Trans. Ind. Inf.*, vol. 13, no. 2, pp. 532–541, Apr. 2017.

Qingqiang Sun received the B.Eng. degree in electrical engineering and automation from Xiamen University, Xiamen, China, in 2017, and the M.Eng. degree from the Department of Control Science and Engineering, Zhejiang University, Hangzhou, China, in 2020.

His research interests include data-based modeling, process data deep learning, and soft sensing.

Zhiqiang Ge (Senior Member, IEEE) received the B.Eng. and Ph.D. degrees in automation from the Department of Control Science and Engineering, Zhejiang University, Hangzhou, China, in 2004 and 2009, respectively.

He was a Research Associate with the Department of Chemical and Biomolecular Engineering, Hong Kong University of Science Technology, Hong Kong, from July 2010 to December 2011 and a Visiting Professor with the Department of Chemical and Materials Engineering, University of Alberta, Edmonton, AB, Canada, from January 2013 to May 2013. He was an Alexander von Humboldt Research Fellow with the University of Duisburg–Essen, Duisburg, Germany, from November 2014 to January 2017, and also a JSPS Invitation Fellow with Kyoto University, Kyoto, Japan, from January 2018 to August 2018. He is currently a Full Professor with the College of Control Science and Engineering, Zhejiang University. His research interests include industrial big data, process monitoring, soft sensor, data-driven modeling, machine intelligence, and knowledge automation.