A Data-Driven Self-Supervised LSTM-DeepFM Model for Industrial Soft Sensor

Lei Ren, *Member, IEEE*, Tao Wang, Yuanjun Laili, *Member, IEEE*, and Lin Zhang, *Senior Member, IEEE*

***Abstract*—Soft sensor, as an important paradigm for industrial intelligence, is widely used in industrial production to achieve efficient monitoring and prediction of production status including product quality. Data-driven soft sensor methods have attracted attention, which still have challenges because of complex industrial data with diverse characteristics, nonlinear relationships, and massive unlabeled samples. In this article, a data-driven self-supervised long short-term memory–deep factorization machine (LSTM-DeepFM) model is proposed for industrial soft sensor, in which a framework mainly including pretraining and finetuning stages is proposed to explore diverse industrial data characteristics. In the pretraining stage, an LSTM-autoencoder is first unsupervised pretrained. Then, based on two self-supervised mask strategies, LSTM-deep can explore the interdependencies between features as well as the dynamic fluctuation in time series. In the finetuning stage, relying on pretrained representation, the temporal, high-dimensional, and low-dimensional features can be extracted from the LSTM, deep, and FM components, respectively. Finally, experiments on the real-world mining dataset demonstrate that the proposed method achieves state of the art comparing with stacked autoencoder-based models, variational autoencoder-based models, semisupervised parallel DeepFM, etc.**

***Index Terms*—Deep learning, industrial big data, industrial intelligence, product quality prediction, self-supervised learning, soft sensor.**

I. Introduction

PRODUCTION status prediction, such as product quality prediction, is critical for high-quality product delivery and core competencies [1]. In the mining and chemical industries, fast and effective prediction provides engineers with information to take early action to control state variables, then further improving product quality [2]. However, traditional methods of product quality prediction, relying on offline laboratory analysis, are generally untimely. In recent years, soft sensors [3] have been widely used to estimate critical quality variables that are not measurable in industrial processes.

Soft sensors combine hardware sensors and computer programs, thus enabling real-time prediction and cost reduction [4]. Approaches of soft sensors can be mainly divided into model-driven and data-driven methods. Model-driven approaches are mainly based on the theoretical hypotheses and experimental tests. And they are often difficult and time-consuming, hence are not suitable for complex industrial production systems [5]. In contrast, by modeling with easily measurable process variables and laboratory quality analysis results, data-driven approaches can provide a reliable and stable online estimation. Therefore, data-driven approaches are widely used in complex situations.

However, due to the complex characteristics of industrial data, data-driven approaches usually have difficulties in modeling. For example, froth flotation is an important industrial process in the mining industry. It uses the hydrophilic differences between minerals and impurities to purify useful minerals. However, owing to complex physiochemical reaction mechanisms, industrial process data are naturally low quality and high noise with high nonlinear temporal correlations. Moreover, due to the inefficient laboratory analysis, the sampling rate of quality variable is lower than that of process variables. Thus, the unlabeled data are abundant and usually much more than the labeled data. These problems greatly limit the application of soft sensors in industrial processes.

Fortunately, several methods have been proposed to deal with the large amounts of unlabeled data as well as complex industrial data feature modeling problems. For example, co-training [6] is a popular semisupervised learning algorithm that can augment the unlabeled samples with multiple training iterations. Besides, methods based on generative models [7], [8] are also commonly used to solve the lack of labeled samples. In [9], the variational autoencoder (VAE)-based methods fit the data to a finite-dimensional mixed Gaussian distribution. Thus, artificial samples can be obtained by randomly sampling from the distribution. In addition, to cope with the complex data characteristics, an improved long short-term memory (LSTM) with the VAE [10] is designed to extract the complex temporal information in the industrial time series, whereas semisupervised parallel DeepFM (SS-PdeepFM) [11] is used to extract low-dimensional and high-dimensional features in a single time

Manuscript received April 28, 2021; revised July 26, 2021; accepted November 17, 2021. Date of publication November 30, 2021; date of current version June 13, 2022. This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFB1703903 and in part by the National Science Foundation of China under Project 92167108, Project 62173023, and Project 61836001. Paper no. TII-21-1851. *(Corresponding author: Lei Ren.)*

The authors are with the School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China (e-mail: renlei@buaa.edu.cn; owntaz@buaa.edu.cn; lailiyuanjun@buaa.edu.cn; zhanglin@buaa.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TII.2021.3131471.

Digital Object Identifier 10.1109/TII.2021.3131471

step. In summary, there are still several major challenges with existing data-driven approaches, which are as follows.

First, many current models have been proposed to deal with a wide range of industrial data characteristics (e.g., temporal correlations of data sequences, low quality and high noise, and large amounts of unlabeled data). However, the abovementioned models only analyze and model a specific characteristic of industrial data, and cannot achieve fusion learning of various industrial data characteristics. Especially, since the industrial structures become increasingly complicated, there is still lacking a systematic soft sensor framework for complex industrial process prediction with diverse data characteristics.

Second, most existing models are usually modeled from time series or a single time step. It is crucial to consider the dynamic temporal correlations across all time steps as well as the intrinsic nonlinear relationships of the key indicator and process variables at each time step. However, there are few models that consider these two relationships at the same time.

Third, some semisupervised learning methods (e.g., co-training and graph-based methods) are not suitable for noisy data or few labeled samples scenes. And some methods, such as traditional autoencoders, can only reconstruct the input but lack effective mining of interdependent relationships of features and time series.

To solve the abovementioned problems, a data-driven self-supervised LSTM-deep factorization machine (DeepFM) model is proposed for industrial soft sensor prediction with the main contributions given as follows.

1) A new systematic soft sensor framework is proposed for complex industrial process prediction, which includes a method for processing the data sequences. This method is based on the representation learning of massive process data to deal with industrial data noise. Meanwhile, it utilizes self-supervised learning to effectively extract the useful information hidden in a large amount of unlabeled data. Therefore, the fusion learning of various industrial data characteristics can be achieved well.
2) An LSTM-DeepFM structure is proposed for multifeature extraction in complex environment. The LSTM-autoencoder can extract temporal information in industrial data. By fusing this information, DeepFM can better reveal the implicit correlations between the process variables and the key indicator.
3) A novel self-supervised learning method is proposed for multidomain feature mining of potential multidimensional information. Based on two mask strategies, this method can explore the interdependencies between features as well as the dynamic fluctuation in time series. Thus, the information of unlabeled samples can be fully mined.

Finally, to demonstrate the performance of the proposed self-supervised LSTM-DeepFM model, it is applied to an industrial froth flotation process to predict the residual impurity in the purified mineral.

The rest of this article organized as follows. In Section II, the brief review about the data-driven soft sensor, LSTM, and DeepFM is presented. Then, in Section III, the overview of the proposed framework, including data processing, pretraining stage, and finetuning stage, is proposed. Methodology and its corresponding case study are presented in Sections IV and V, respectively. Finally, Section VI concludes this article.

II. Preliminaries

In this section, the recent advancements of the data-driven soft sensor and the deep feature extraction methods are presented.

A. Data-Driven Soft Sensor

Recently, data-driven soft sensor techniques have been widely developed for specific industrial applications. For example, a deep probabilistic transfer learning framework [12] and a model-agnostic metalearning method [13] were proposed to improve the model performance when migrating from the relevant source domain to the target domain. In addition, to address the LSTM's inadequacies, a VAE-based LSTM was designed, which adopts batch training and L2 regularization techniques to learn crucial data information from various process data [10]. In [14], another improved method of LSTM called gated convolutional neural network-based transformer (GCT) was implemented to deal with the gradient vanishing and the parallel computing difficulties. Also, overfitting is a problem that is easily neglected in soft sensor modeling. The bound optimization theory and variational Bayesian inference were integrated in [15] to regularize the extreme learning machine, and satisfying performance was obtained in energy-efficient building design experiments.

B. Deep Feature Extraction

Deep feature extraction can be accomplished using a variety of approaches, with the stacked autoencoder (SAE) being one of the most widely used backbone networks. Variablewise weighted SAE (VW-SAE) [16], for example, is a hierarchical VW-SAE that extracts output-related feature representation. In addition, gated stacked target-related autoencoder (GSTAE) [17] is utilized to extract different levels of abstract features, with gate neurons governing the flow of information in different layers. Besides, slow and multidimensional feature extractions are also crucial in industrial processes. A Siamese network, called supervised slow feature analysis Siamese network (SSFAN) [18], extracts the latent features from the time series based on temporal slowness aspect, whereas SS-PdeepFM [11] extracts low- and high-dimensional features in a single time step. However, the SAE-based models may trivially copy their inputs to outputs without finding useful patterns in the data. And the advantages of Siamese network and SS-PdeepFM are complementary in the complete process of feature extraction, but they cannot perform well enough on their own.

C. Long Short-Term Memory-Deep Factorization Machine

In this part, the brief introduction about LSTM-DeepFM, including the LSTM and DeepFM, is presented.

1) Long Short-Term Memory: LSTM is a popular network for extracting temporal features in industrial time series. There

are three gate controllers in an LSTM unit: input gate, forget gate, and output gate. The input gate identifies important inputs, while the forget gate learns to preserve information to long-term memory. And the output gate can control the output of the current time step. Assuming that the input sequences are $\{x_1^c, x_2^c, \ldots, x_T{}^c\}$, the forward pass of an LSTM unit at time step t is as follows:

$$i_t = \sigma\left(W_{ix}x_t{}^c + W_{ih}h_{t-1} + b_i\right) \tag{1}$$

$$f_t = \sigma\left(W_{fx}x_t{}^c + W_{fh}h_{t-1} + b_f\right) \tag{2}$$

$$o_t = \sigma\left(W_{ox}x_t{}^c + W_{oh}h_{t-1} + b_o\right) \tag{3}$$

$$\tilde{c}_t = \tanh\left(W_{cx}x_t{}^c + W_{ch}h_{t-1} + b_c\right) \tag{4}$$

$$m_t = f_t \odot m_{t-1} + i_t \odot \tilde{c}_t \tag{5}$$

$$h_t = o_t \odot \tanh\left(m_t\right) \tag{6}$$

where i_t, f_t, and o_t are the input gate, forget gate and output gate, respectively; $\tilde{c}_t$ is the intermediate state, m_t is a memory cell, and h_t is the hidden state; tanh and σ are nonlinear activation functions; and $\odot$ denotes pointwise multiplication.

2) Deep Factorization Machine: DeepFM is often used in recommendation systems to extract low- and high-dimensional features in the data. It consists of two parts, FM component and deep component.

1) *FM component:* The core idea of FM is to do pairwise matrix decomposition between features. Compared with the linear model, it can mine the influence of the second-order combination of features. The objective function of the FM is shown in (7). The w_i and the w_0 are the weight and bias of a linear regression. x_i is the ith variable and n is the number of variables. $< \boldsymbol{v_i}, \boldsymbol{v_j} >$ denotes the weight of the second-order feature combination.

$$y_{\text{fm}} = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} < \boldsymbol{v_i}, \boldsymbol{v_j} > x_i x_j \tag{7}$$

$$\begin{aligned}
&\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} < \boldsymbol{v_i}, \boldsymbol{v_j} > x_i x_j \\
&= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} < \boldsymbol{v_i}, \boldsymbol{v_j} > x_i x_j - \frac{1}{2}\sum_{i=1}^{n} < \boldsymbol{v_i}, \boldsymbol{v_i} > x_i x_i \\
&= \frac{1}{2}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{f=1}^{k} v_{i,f} v_{j,f} x_i x_j - \sum_{i=1}^{n}\sum_{f=1}^{k} v_{i,f} v_{i,f} x_i x_i\right) \\
&= \frac{1}{2}\sum_{f=1}^{k}\left(\left(\sum_{i=1}^{n} v_{i,f} x_i\right)^2 - \sum_{i=1}^{n} v_{i,f}^2 x_i^2\right).
\end{aligned} \tag{8}$$

For each variable, the FM component learns a 1-D vector of fixed size k by the embedding layer. Therefore, the weight between the variables x_i and x_j can be expressed by the inner product of the feature corresponding vectors v_i and v_j. The complexity of the original FM is $O(k \cdot n^2)$. By rewriting the mathematical formula, as shown in (8), it can become the $O(k \cdot n)$ complexity, so it is very suitable for the industrial scene.

2) *Deep component:* The deep component is a deep neural network (DNN) that can capture high-order features, accepting the normalized features as input. The forward propagation of DNN is shown as follows:

$$a^{(l+1)} = \text{Activation}\left(W^{(l)}a^{(l)} + b^{(l)}\right) \tag{9}$$

where $a^{(l)}$ is the output of the lth hidden layer, and $W^{(l)}$ and $b^{(l)}$ are the weight and bias, respectively. The activation function is the parametric rectified linear unit (PReLU) [19]

$$\text{PReLU}(x) = \max(0, x) + \alpha * \min(0, x) \tag{10}$$

where α is a learnable parameter.

III. Overview of the Proposed Framework

In this section, as shown in Fig. 2, the overview of the proposed framework, mainly including data processing, pretraining stage, and finetuning stage, is presented.

In the data processing part, raw time series are collected from soft sensors in the industrial processes, and then some features of the raw time series are discretized to effectively suppress data noise. Also, the different value ranges of process variables are solved by data normalization. Moreover, feature selection selects more representative, independent, and quality-related features for modeling. Therefore, quality information can be better extracted from the data.

In the pretraining stage, the LSTM-autoencoder is first unsupervised pretrained to obtained the latent embedding. The latent embedding and the continuous input features are concatenated and masked. Then, they are normalized and sent to the deep component autoencoder. The self-supervised pretraining loss function is the masked features and time series prediction loss. The whole pretraining process is depicted in Fig. 1(a).

In the finetuning stage, the pretrained LSTM-encoder is employed to extract the temporal features from the continuous features. Subsequently, the continuous features concatenated with the latent embedding are normalized and directly fed into the pretrained deep component encoder. So high-dimensional features can be explored. In parallel, the discrete features are sent into the embedding layer and FM component for low-dimensional features extraction. Finally, the comprehensive output is built based on the two kinds of outputs. Fig. 1(b) shows the whole finetuning process of the proposed framework.

IV. Methodology

In this section, data processing technologies, LSTM-DeepFM structure, pretraining stage, and finetuning stage will be discussed.

A. Data Processing Technologies

First, we discuss the data processing technologies in this part.

1) Data Binning: One of the inevitable problems in industrial data analysis is data noise. It may degrade the performance of the

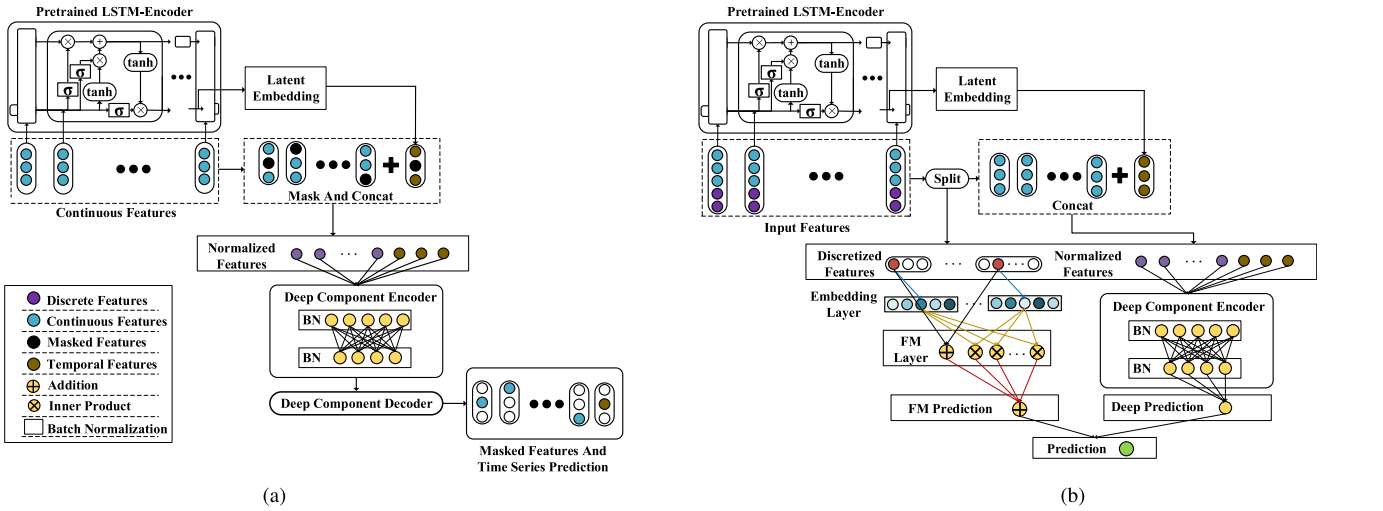


Fig. 1. Detailed structures of the model during self-supervised pretraining stage and supervised finetuning stage. (a) self-supervised pre-training LSTM-deep model. (b) supervised Fine-tuning LSTM-deepFM model.

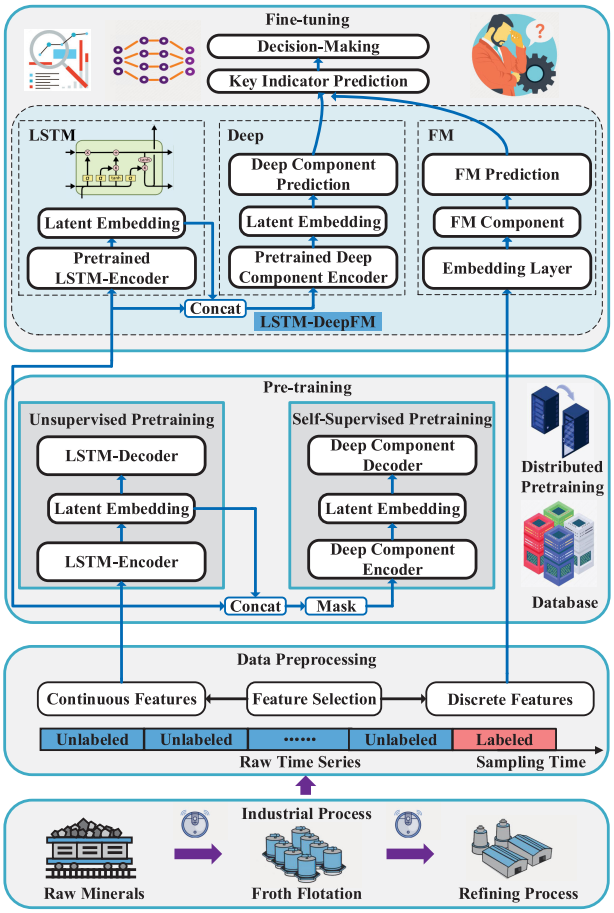


Fig. 2. Framework of the proposed approach.

model. Therefore, a k-means-based binning method is applied to suppress the data noise effectively. It ensures that all values in each bin have the same nearest center of a 1-D k-means cluster.

In order to select a reasonable number of bins and improve the performance, silhouette coefficient [20] is applied as follows:

$$\mathrm{Sil}(i) = \frac{(b(i) - a(i))}{\max(b(i), a(i))} \tag{11}$$

where $a(i)$ is the mean of the distances between the sample to the others, and $b(i)$ is the mean nearest cluster distance, that is the mean instance to the instances of the next closest cluster. Silhouette coefficient closes to +1 means that the instance is well inside its own cluster and far from other clusters.

2) Feature Selection: Feature selection refers to select a subset of features according to their feature importance. Commonly used techniques are Lasso regularization [21] and tree-based feature selection [22]. However, there are usually complex relations between industrial data. For any feature with large variance, the tree-based model can always find the split point to optimize the loss function, which causes a deviation in the ranking of feature importance.

To correct the feature importance bias, permutation importance (PIMP) [23] is applied to soft sensor modeling, and this method can be clearly elucidated in Algorithm 1.

B. Pretraining Stage

Here, we introduce two fundamental methods employed in the pretraining stage.

1) LSTM-Autoencoder: As mentioned in Section I, the relationships among the process variables are very complex. The temporal and nontemporal features will influence the quality index. Therefore, we reconstruct the input sequences by an LSTM-autoencoder to obtain the temporal features from the latent representation, with which the DeepFM can better predict the quality variable.

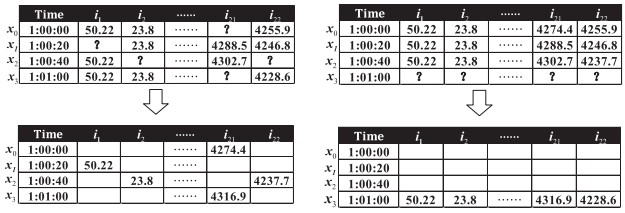
The LSTM-autoencoder consists of two LSTMs: an encoder and a decoder. The encoder will translate the input sequence into

Algorithm 1 PIMP for feature selection

Input: The labeled dataset X_i, y_i; a model with ranking feature importance; a threshold; the number of model runs n.

Output: The subset of the features;

1: Create the null importance distribution based on n model runs on a shuffled version of the target.
2: Fit the model on the original target and gather the feature importance.
3: For each feature, if the actual importance is larger than the threshold of the null importance distribution: leave the feature.
4: **return** The selected features.



(a) (b)

Fig. 3. Two tasks of self-supervised learning. For the sake of simplicity, the latent representation is omitted in (a) and (b). Detailed structure of the model in the self-supervised pretraining stage is shown in Fig. 1(a). (a) Self-supervised learning task 1. (b) Self-supervised learning task 2.

a reduced-dimensional feature vector e_1 (latent embedding)

$$e_1 = \text{LSTMEncoder}(x^c) \tag{12}$$

where x^c is the continuous input features of the input sequence. The decoder will reconstruct the original input sequence by decoding the latent embedding from the encoder

$$x^{c'} = \text{LSTMDecoder}(e_1). \tag{13}$$

The loss function loss_1 is defined as the mean squared error (mse) between the predicted sequence $x^{c'}$ and the real sequence x^c

$$\text{loss}_1 = \frac{1}{B}\sum_{b=1}^{B}\sum_{j=1}^{T}||x_{b,j}^{c'} - x_{b,j}^{c}||^2 \tag{14}$$

where B and T are the batch size and the time step of the LSTM, respectively.

To reduce the loss, the encoder tends to preserve temporal information in the latent embedding as much as possible. Then, the latent representation concatenated with the flattened input sequence is served as the input of the DeepFM

$$s = \text{concat}(\text{flatten}(x^c), e_1). \tag{15}$$

2) Self-Supervised Learning: Self-supervised learning uses pretext to mine its own supervision information from massive unsupervised data and trains the network through the constructed supervision information. Therefore, the model can learn more effective representation through self-supervised training objective. In this article, as shown in Fig. 3, two tasks of self-supervised learning are proposed to mine the information of raw data from two domains. Task 1 seeks to predict missing feature columns regardless of chronological order, so the model can mine the interdependencies between features. And task 2 seeks to predict missing time series to capture the chronological relationships, so the dynamic fluctuation in time series can also be taken into consideration. Fig. 3(a) and (b) illustrates self-supervised learning task 1 and task 2, respectively.

For a mathematical form, considering a binary mask $M \in \{0,1\}^{B\times D}$, the deep component encodes the input $(1-M)\cdot s$ and decodes the reconstructed features $M \cdot \hat{s}$, as shown in the following:

$$e_2 = \text{Deep Component Encoder}((1-M)\cdot s) \tag{16}$$

$$\hat{s} = \text{Deep Component Decoder}(e_2). \tag{17}$$

In the self-supervised phase, the decoder's last full-connected (FC) layer is multiplied by M because only the unknown features are considered. The self-supervised training loss_2 is defined as follows:

$$\text{loss}_2 = \frac{1}{B}\sum_{b=1}^{B}\sum_{j=1}^{D}\left|\frac{(\hat{s}_{b,j} - s_{b,j})\cdot M_{b,j}}{\sqrt{\sum_{b=1}^{B}\left(s_{b,j} - \frac{1}{B}\sum_{b=1}^{B} s_{b,j}\right)^2}}\right|^2 \tag{18}$$

As the industrial process variables may have different ranges, the normalization term with population standard deviation is added to the loss_2. At each iteration, the $M_{b,j}$ is sampled independently from a Bernoulli distribution with parameter p.

C. Finetuning Stage

During the finetuning stage, the binary mask M is removed, and an embedding layer and an FM component are added.

1) Embedding Layer: Due to k-means discretization, some continuous features are transformed to sparse one-hot variables with different dimensions. The embedding layer is used to convert these one-hot variables into dense vectors with the unified embedding size k. It also acts as an auxiliary part of the FM component

$$v_i = \text{Embedding Layer}\,(x_i)\,. \tag{19}$$

In the finetuning stage, the (7) can be optimized by training the embedding layer. And it can be simplified in the following, where x^d is the discrete features.

$$y_{fm} = \text{FM Component}\,(x^d)\,. \tag{20}$$

During the finetuning stage, the binary mask M is removed. The output of the deep component is shown in (21). And the only new parameters introduced in the deep component are the regression layer weights $W \in \mathbb{R}^{1\times H}$, where H is the hidden size of the last FC layer in the deep component encoder.

$$e_2 = \text{Deep Component Encoder}(s) \tag{21}$$

$$y_{\text{deep}} = We_2. \tag{22}$$

2) Long Short-Term Memory-Deep Factorization Machine: The model structure of LSTM-DeepFM in the finetuning stage

is shown in Fig. 1(b). In the original DeepFM model, the FM component and the deep component share the same embedding layer. The discrete and the normalized features are both transformed to vectors of the same size, which may bring more computing costs. Also, in order to make the deep component benefit from the self-supervised training, the parameter sharing of the FM component and the deep component is removed. Therefore, the FM component and the deep component can be trained individually.

In our implementation, we only feed the discrete variables into the FM component. The deep component takes the normalized features as input. Finally, the comprehensive output of LSTM-DeepFM is based on both of the two components

$$y_{\text{LSTM-DeepFM}} = \beta_1 y_{\text{deep}} + \beta_2 y_{\text{fm}} \tag{23}$$

where β_1 and β_2 are both initialized to 0.5 and set as trainable parameters in consider of the different contributions of the two components. For example, assuming the supervised loss function in the finetuning stage is $\mathcal{L}$, which is also a function of $\hat{y}$, then the stochastic gradient descent can be utilized to optimize β_1 and β_2 as follows:

$$\beta_1 = \beta_1 - \text{lr} * \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \beta_1} = \beta_1 - \frac{\partial \mathcal{L}}{\partial \hat{y}} * \text{lr} * y_{\text{deep}} \tag{24}$$

$$\beta_2 = \beta_2 - \text{lr} * \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \beta_2} = \beta_2 - \frac{\partial \mathcal{L}}{\partial \hat{y}} * \text{lr} * y_{\text{fm}} \tag{25}$$

where lr is the learning rate. After many trials, the average values of β_1 and β_2 are approximately 1.64 and 1.41, respectively, which proves that both components play equally important roles in the soft sensor prediction.

In summary, the finetuning stage is relatively inexpensive compared to the pretraining stage. In the pretraining stage, massive labeled and unlabeled samples are used for computing self-supervised training objective function. However, in the finetuning stage, only the labeled samples are used. Through the unsupervised representation learning by masked self-supervised learning task, the model can converge faster and has a better generalization performance in the finetuning stage. In this way, the labeled and unlabeled samples can be fully utilized so that more quality-related information can be derived.

V. Case Study

To measure the proposed method, a case study is presented to illustrate the advantages of our proposed method over the methods based on generation model and depth feature extraction.

A. Industry Process

In mineral processing, froth flotation is a purification process for separating valuable minerals from waste gangue by exploiting their different hydrophobicity. A valuable mineral is more hydrophobic than the waste gangue because of the employment of surfactants and other chemical reagents. Depending on the selective adhesion of air bubbles to mineral surface, the air bubbles will attach to more hydrophobic particles.

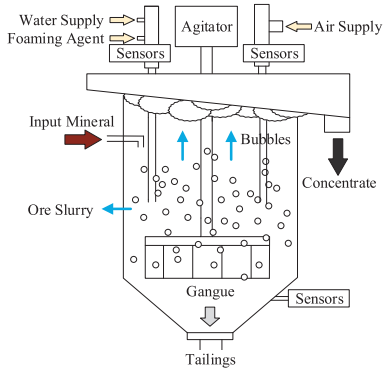


Fig. 4. Schematic froth flotation cell.

TABLE I
DESCRIOTIONS OF THE RECORDED FEATURES

No	Sampling Frequency	Description
i_1	Hourly	Purity of mineral (%) before purification
i_2	Hourly	Impurity content (%) before purification
i_3	Hourly	Impurity content (%) with 2 hours delay
$i_4 \sim i_5$	20 seconds	Speed of each reagent flow (m^3/h)
i_6	20 seconds	The speed of ore slurry flow (t/h)
i_7	20 seconds	The PH of ore slurry
i_8	20 seconds	The density of ore slurry (kg/cm^3)
$i_9 \sim i_{15}$	20 seconds	Air flow speed in each column (Nm^3/h)
$i_{16} \sim i_{22}$	20 seconds	Froth level in each column (mm)
o_1	Hourly	Impurity content (%) after purification

Before froth flotation, the ore needs to be crushed and ground into fine particles (the particle sizes are typically less than 0.1 mm). Then, by mixing the particles with water, ore slurry is formed. This process is called as liberation. Next, the slurry will be treated with some chemical agents, including surfactants and frothers. Accompanied by the air supply and frothers, the slurry is agitated by the agitator to generate bubbles, and more hydrophobic particles (valuable minerals) will stick to the air bubbles. Then, the bubbles float up to the top of slurry, hence forming froth. Finally, the attached valuable minerals will be collected and concentrated, and sent to a further process for refining. The whole process is shown in Fig. 4.

In the froth flotation cell, different process variables are measured by sensors. The valuable mineral and impurity were sampled hourly, and the laboratory analysis of purity(%) and impurity(%) will take 2 h. The other process variables including the speed of reagent flow and the pH of ore slurry were measured every 20 s. Thus, the ratio of labeled to unlabeled samples is 1:179. The proportion of the residual impurity in the purified mineral was considered as the quality index to evaluate the purification process.

In this article, the utilized dataset is obtained from real flotation processes. There is a total of 3670 labeled samples. Each sample has 22 process variables and a label from laboratory analysis. A total of 3000 samples of the dataset are selected as the training set, and the other 670 samples collected in the last month are utilized as the test set. The information about all variables are shown in Table I. After data binning, data normalization, and

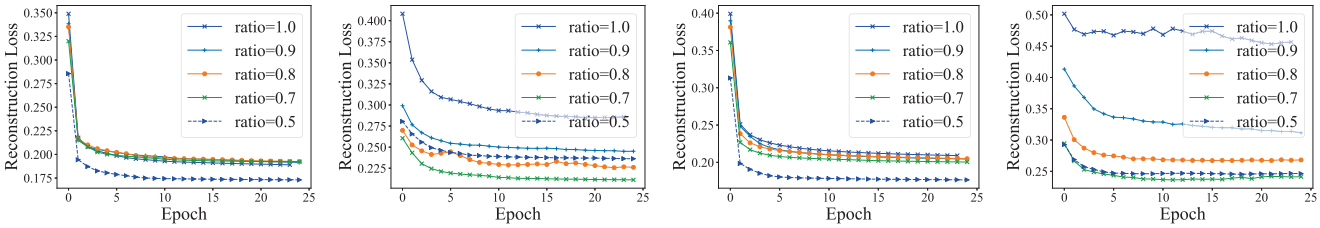


Fig. 5. Comparison of different number of feature selection ratios and embedding size in the experiment. (a) Train Loss(embed = 24). (b) Validation Loss(embed = 24). (c) Train Loss(embed = 12). (d) Validation Loss(embed = 12).

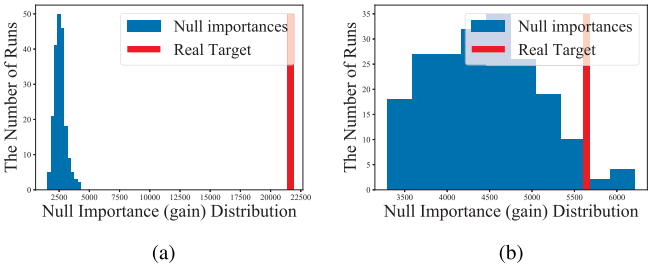


Fig. 6. Split importance is the number of times the feature is used in the LGB model. (a) Split Importance of i_1. (b) Split Importance of i_{10}

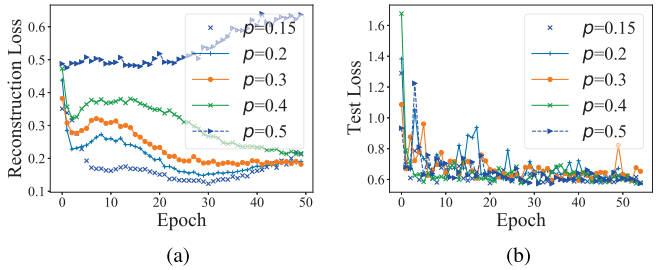


Fig. 7. Reconstruction loss (a) and the test loss (mse) with different mask rate settings in the pretraining and the finetuning stage, respectively. (a) Pretraining stage. (b) Finetuning stage.

feature selection, each sample contains a 4-D discretized vector, an 11-D normalized vector, and a 1-D label.

B. Experiment

In this section, experiments are carried out to test the effectiveness of the proposed methods. The experiment is mainly divided into two parts, containing the hyperparameters selection and comparison with other methods.

1) Feature Selection: In the PIMP test, the lightgbm (LGB) [24] was chosen as the base model, and the number of model runs was set as 200. As shown in Fig. 6, the distance between the red line and the blue distribution measures the effectiveness of the feature. In Fig. 6(b), the actual importance is inside the null importance distribution, which means that the feature may not be effective.

The threshold set was $\{1, 0.9, 0.8, 0.7, 0.5\}$, and the dimensions of latent embedding were set to 12 and 24 in the experiment. As shown in Fig. 5, although selecting a small number of features can obtain good results in the training set, the model does not perform as well in the validation set. It can be concluded that feature selection effectively narrows the search place of the training process and accelerates the convergence of the model. However, too few features can damage the robustness of the model. Based on these experiments, only the top 70% of the features are retained, thus preventing the model from reducing the generalization ability of unknown data.

2) Self-Supervised Learning: To determine the value of mask rate p, a relatively optimal number of features and latent dimension were chosen. As shown in Fig. 7, the convergence of the model had been tested in different mask rate settings.

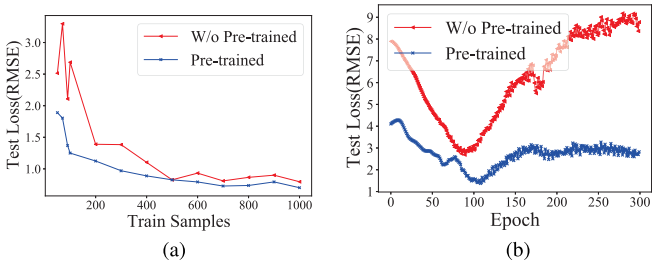


Fig. 8. (a) Test RMSE of the model with and without pretrained under partially labeled samples. (b) Training curve of the model with and without pretrained under 100 labeled samples. (a) RMSE of partial labeled samples. (b) Training Curves of 100 samples.

It can be concluded that a small p makes the model converge faster, whereas a large p leads to convergence difficulties in the pretraining stage and loss fluctuations in the finetuning stage, as shown in Fig. 7(b).

In terms of interpretability, self-supervised learning can mine the interdependencies between features. For example, the pH of ore slurry can be guessed from the flow rates of the slurry and reagents, while the purity of minerals and major impurity components are also generally intrinsically related. With self-supervised learning, unsupervised representation learning can explore the complex relationships and provide an improved encoder for the supervised learning task.

As shown in Fig. 8(a), to train the LSTM-DeepFM, the number of labeled samples increases 100 each time from 100 to 1000. The pretrained model always performs better than the model with random initialization. Moreover, in Fig. 8(b), the

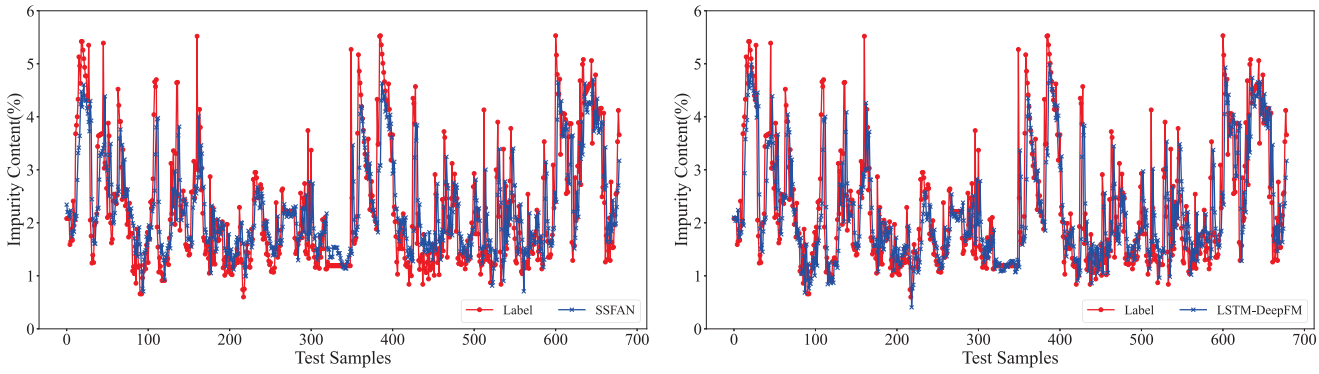


Fig. 9. Comparison between SSFAN and the LSTM-DeepFM. For simplicity, not all the fitting plots are shown here. The SVR and LGB were only trained with the original labeled samples. The kernel function of the SVR is radial basis, and the penalty parameter of its error term is set to 0.01. The hyperparameters of LGB were selected according to [24]. For VAE-WGAN, VAE-NN, and SS-PdeepFM, each model was trained with a mixture of original labeled samples and artificial samples. The size of the original labeled samples and the artificial samples are both 3000. The size of the hidden feature was 20. In addition, the encoder, decoder, and discriminator of each model were DNNs with two hidden layers of (256, 64) units. The other hyperparameters were from [7], [8], and [11]. For SSFAN, the network has four hidden layers of (256, 128, 32, 16) units and the number of slow features is set to 32. Besides, GSTAE has three hidden layers of (256, 128, 64) units, and the label is normalized to [0, 1] to meet the output requirements of GSTAE. The design of the model structure and hyperparameters were taken from [17] and [18], and the number of neurons was scaled up in order to enhance the model performance. (a) SSFAN. (b) LSTM-DeepFM.

TABLE II
DETERMINED VALUES OF THE HYPERPARAMETERS

Hyperparameter	p	t	e	$thres$
Value	0.15	4	24	70%

TABLE III
PERFORMANCE ON DIFFERENT METHODS

	Test Set	
Model	RMSE	MAE
SVR	0.8315 ± /	0.6006 ± /
LGB	0.8091 ± 0.0092	0.6083 ± 0.0118
VAE-WGAN	0.7985 ± 0.0116	0.5847 ± 0.0164
VAE-NN	0.7806 ± 0.0063	0.5454 ± 0.0085
GSTAE	0.7735 ± 0.0068	0.5296 ± 0.0097
SS-PdeepFM	0.7650 ± 0.0036	0.5206 ± 0.0073
SSFAN	0.7571 ± 0.0033	0.5235 ± 0.0092
LSTM-DeepFM	**0.7497 ± 0.0026**	**0.5091 ± 0.0071**

The bold values represent the root mean squared error and the mean absolute error of our proposed method LSTM-DeepFM.

pretrained model is more stable and not easy to be overfitting, even in the small labeled sample scenarios.

Then, the hyperparameters are determined as followed. First, the mask rate p is set to 0.15 so that the training curve is more stable in the finetuning stage. Second, the time step of the input sequence t is fixed to 4. In addition, based on the aforementioned results, the latent embedding size e and the threshold thres of the PIMP are fixed to 24% and 70%, respectively. Other parameters are determined as their empirical values. For example, the hidden layers of the deep component are set to (256, 64) neural units. The determined hyperparameters are listed in Table II.

3) Evaluation: Some other experiments are conducted to verify the effectiveness of the model in soft sensor modeling. Support vector regression (SVR), LGB, variational autoencoder and neural network (VAE-NN), variational autoencoder and Wasserstein GAN (VA-WGAN), SS-PdeepFM, SSFAN, and GSTAE are used for comparison. Among them, SVR [25] is a more traditional and effective method. It is widely used in the industrial field and has good interpretability. LGB is an ensemble tree-based model that performs fairly well in some data science competitions. Furthermore, VAE-NN and VAE-WGAN are two methods based on generative models, which are commonly used to augment labeled samples. In addition, SS-PdeepFM, SSFAN, and GSTAE represent different soft sensor methods for dynamic feature extraction and modeling. To evaluate the performance of different models, the root-mean-squared error (RMSE) and the mean absolute error (MAE) are used as basic metrics. Results of the different models are shown in Table III and Fig. 9. All results reported are the mean and std. of 20 runs. The source code of LSTM-DeepFM is available on GitHub.[1]

In these experiments, the SVR and LGB are used as baseline models. However, due to the small number of labeled samples, they do not perform well. The VAE-WGAN and VAE-NN solve the labeled data paucity to some extent. They generate a lot of artificial samples that can effectively improve the accuracy of the model. However, these two models are more suitable for the cases of small unlabeled samples. And compared with the unsupervised representation learning of LSTM-DeepFM, the generative model does not transfer the knowledge learned from data to the posttraining model. So it's not practical under the context of our article.

Both of the SS-PdeepFM and LSTM-DeepFM get decent performance in general. The SS-PdeepFM uses a label broadcasting method, so the samples are augmented to some extent. On the one hand, the label will only broadcast to few unlabeled samples so that lots of unlabeled samples are still wasted. On the other hand, there also exists temporal quality-related information that is

[1][Online]. Available: https://github.com/iamownt/LSTM-DeepFM

TABLE IV
ABLATION STUDY

		Test Set			
#	Model	RMSE	MAE	Complexity	Time
1	SS-PdeepFM	0.7650 ± 0.0036	0.5206 ± 0.0073	$O(l \cdot d^2 + k \cdot n_c)$	0.3069 ms
2	**LSTM-DeepFM**	**0.7497 ± 0.0026**	**0.5091 ± 0.0071**	$O(t \cdot h^2 + l \cdot d^2 + k \cdot n_c)$	0.6650 ms
3	- LSTM	0.7533 ± 0.0030	0.5119 ± 0.0079	$O(l \cdot d^2 + k \cdot n_c)$	0.3456 ms
4	- SSL	0.7599 ± 0.0059	0.5175 ± 0.0085	$O(t \cdot h^2 + l \cdot d^2 + k \cdot n_c)$	0.6650 ms
5	- Feature Selection	0.7518 ± 0.0041	0.5102 ± 0.0084	$O(t \cdot h^2 + l \cdot d^2 + k \cdot n_c)$	0.7151 ms
6	- DFM	0.7714 ± 0.0050	0.5289 ± 0.0062	$O(t \cdot h^2)$	0.2980 ms

SSL denotes the self-supervised learning. All results reported are the mean of 20 runs. Note that in row 6, if DFM is removed from LSTM-DeepFM, it will degenerate to a simple LSTM-autoencoder, so it is slightly worse than SS-PdeepFM.
The bold values represent the root mean squared error and the mean absolute error of our proposed method LSTM-DeepFM.

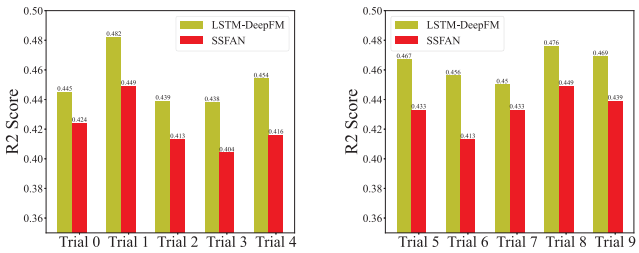


Fig. 10. Trial 0 to Trial 4 represent five random equal divisions of the training and test sets. Trial 5 to Trial 9 represent the exchange of the training and test sets corresponding to the first five trials.

TABLE V
COMPLEXITY ANALYSIS AND COMPUTATION COST PER SAMPLE

Method	Computational complexity	Computation cost per sample
LSTM	$O(t \cdot h^2)$	0.3053 ms
VAE-WGAN	$O(l \cdot d^2)$	0.1016 ms
VAE-NN	$O(l \cdot d^2)$	0.1016 ms
GSTAE	$O(l \cdot d^2)$	0.8913 ms
SS-PdeepFM	$O(l \cdot d^2 + k \cdot n_c)$	0.3069 ms
SSFAN	$O(l \cdot d^2)$	0.2549 ms
LSTM-DeepFM	$O(t \cdot h^2 + l \cdot d^2 + k \cdot n_c)$	0.6650 ms

important for product quality prediction. This is often ignored by the DNN models. Therefore, the LSTM-DeepFM outperforms the SS-PdeepFM by the temporal features extraction part and the self-supervised training objective.

When comparing the dynamic feature extraction performance of SSFAN, GSTAE, and LSTM-DeepFM, GSTAE performs slightly worse. This is because, unlike LSTM-DeepFM, it does not consider dynamical temporal features while constructing the final output through different levels of abstract representation. In addition, GSTAE is pretrained layer-by-layer with a simple reconstruction of the input. Compared with the dynamic mask strategy, it may trivially copy its inputs to outputs without exploring the interdependencies between features. Besides, SSFAN extracts slow features from time series, but often ignores the potential information of fast-changing features. Furthermore, in the pretraining process of SSFAN, all samples must be computed at the same time to construct the covariance matrix of extracted features, which makes it inappropriate for large-scale datasets. And pretraining also requires the guidance of labeled samples, so that it is not suitable for our scenario with large amounts of unlabeled samples.

4) Statistical Hypothesis Test: To verify the effectiveness of our proposed method, we adopt a statistical hypothesis test to compare our model with the previous best performing model. A 5×2cv paired t-test [26] is conducted in the original training set with 3000 labeled samples. Fig. 10 illustrates the performance of the two models in ten trials. Since the training and test sets are equally divided in each trial, the performance of the model is particularly important in the case of fewer samples. Due to self-supervised learning, LSTM-DeepFM can maintain a consistent performance improvement in ten trials with an interval of [0.017, 0.043] despite the small training size. By setting the null hypothesis that the two models perform equally and the significant level $\alpha = 0.05$, the t statistic and the pvalue could be calculated. Through calculation, we can get pvalue $= 0.0308$, which is lower than α. Therefore, we can reject the null hypothesis and conclude that the performance of the two models is significantly different.

5) Complexity Analysis: The proposed algorithm is mainly composed of LSTM, DNN, and FM. The complexity contribution of each component is extended in Table IV, and t, h, n, n_c, n_d, k, l, and d_i represent the length of input sequence, hidden size of LSTM, input size, continuous input size, discrete input size, FM embedding size, number of layers of DNN, and the number of neurons in DNN's ith layer, respectively.

Therefore, the complexity of each soft sensor method is shown in Table V. It is worth noting that the computational complexity of DNN is actually $O(n \cdot d_1 + \sum_{i=1}^{l-1} d_i d_{i+1} + d_l \cdot 1)$, and for simplicity, we use $O(l \cdot d^2)$ instead.

The lowest computation cost is VAE-NN and VAE-WGAN models, as they have fewer hidden layers. And due to the deep network structure or recursive nature, SSFAN and LSTM have relatively high computational costs. Since GSTAE adopts tanh and sigmoid activation functions to control the information flow, it has the highest computational cost. The complexity of LSTM-DeepFM is the sum of the three components' complexity, and the computation time is approximately the sum of LSTM and DNN's time.

6) Ablation Studies: Ablation studies are conducted to understand the importance of each design choice. Examining rows 1–4 of Table IV, we can see both self-supervised learning and

LSTM clearly contribute to the superior performance of LSTM-DeepFM. Moreover, if we remove the feature selection part (row 5), the performance clearly drops. In addition, row 6 shows that the DeepFM plays an important role in LSTM-DeepFM.

On the one hand, it is necessary to extract the temporal features and decouple the complex relationships among the industrial process variables. The LSTM-DeepFM is proposed to solve this issue. The LSTM component is used to obtain the temporal features. And the DeepFM uses the FM component to extract the low-order features, and the deep component is utilized to mine the high-order features. Therefore, the LSTM-DeepFM can analyze the quality variable better compared with the other soft sensor models.

On the other hand, it is also very important to mine and utilize the information of unlabeled samples. In the pretraining stage, the model makes full use of the data and learns the high-dimensional representation of the raw data. In the finetuning stage, the model only needs to learn the trained representation through the fully connected layer, which greatly improves the speed and reliability of the model training. Our proposed method not only achieves state of the art (SOTA), but also effectively alleviates the problem of insufficient labeled samples. Therefore, the self-supervised learning is valuable and can be extended to other industrial process analyses.

VI. Conclusion

In this article, a data-driven self-supervised LSTM-DeepFM model was proposed for industrial soft sensor prediction. On the one hand, the LSTM-DeepFM model structure can extract the low-dimensional, high-dimensional, and temporal features in the time series. On the other hand, the self-supervised learning method can explore the interdependencies between features as well as the dynamic fluctuation in time series. Therefore, the fusion learning of various industrial data characteristics can be achieved well. Experiments on real-world froth flotation dataset demonstrate the effectiveness and superior performance of the proposed approach by comparing with SVR, LGB, VAE-WGAN, VAE-NN, GSTAE, SS-PdeepFM, and SSFAN.

At present, the soft sensor model can be utilized to assist flotation plant operators in making better decisions, and operators also need some visual assessment information of froth appearance. Future work should focus on how to maintain process reliability and support continuous improvement, as well as further exploring online model predictive control techniques.

References

[1] D. Wang, J. Liu, and R. Srinivasan, "Data-driven soft sensor approach for quality prediction in a refining process," *IEEE Trans. Ind. Informat.*, vol. 6, no. 1, pp. 11–17, Feb. 2010.

[2] H. Luo, S. Yin, T. Liu, and A. Q. Khan, "A data-driven realization of the control-performance-oriented process monitoring system," *IEEE Trans. Ind. Electron.*, vol. 67, no. 1, pp. 521–530, Jan. 2020.

[3] 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. 10, pp. 1575–1596, 2013.

[4] X. Wang, L. T. Yang, Y. Wang, L. Ren, and M. J. Deen, "ADTT: A highly-efficient distributed tensor-train decomposition method for IIoT Big Data," *IEEE Trans. Ind. Informat.*, vol. 17, no. 3, pp. 1573–1582, Mar. 2021.

[5] K. Wang, R. B. Gopaluni, J. Chen, and Z. Song, "Deep learning of complex batch process data and its application on quality prediction," *IEEE Trans. Ind. Informat.*, vol. 16, no. 12, pp. 7233–7242, Dec. 2020.

[6] C. M. Nguyen, X. Li, R. D. Blanton, and X. Li, "Partial Bayesian co-training for virtual metrology," *IEEE Trans. Ind. Informat.*, vol. 16, no. 5, pp. 2937–2945, May 2020.

[7] X. Wang and H. Liu, "Data supplement for a soft sensor using a new generative model based on a variational autoencoder and Wasserstein GAN," *J. Process Control*, vol. 85, pp. 91–99, 2020.

[8] R. Xie, N. M. Jan, K. Hao, L. Chen, and B. Huang, "Supervised variational autoencoders for soft sensor modeling with missing data," *IEEE Trans. Ind. Informat.*, vol. 16, no. 4, pp. 2820–2828, Apr. 2020.

[9] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. Int. Conf. Learn. Representations*, 2014.

[10] W. Xie, J. Wang, C. Xing, S. Guo, M. Guo, and L. Zhu, "Variational autoencoder bidirectional long and short-term memory neural network soft-sensor model based on batch training strategy," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5325–5334, Aug. 2021.

[11] L. Ren, Z. Meng, X. Wang, L. Zhang, and L. T. Yang, "A data-driven approach of product quality prediction for complex production systems," *IEEE Trans. Ind. Informat.*, vol. 17, no. 9, pp. 6457–6465, Sep. 2021.

[12] Z. Chai, C. Zhao, B. Huang, and H. Chen, "A deep probabilistic transfer learning framework for soft sensor modeling with missing data," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published, doi: 10.1109/TNNLS.2021.3085869.

[13] Y. Lu, X. Peng, D. Yang, M. Yang, and W. Zhong, "Model-agnostic meta-learning with optimal alternative scaling value and its application to industrial soft sensing," *IEEE Trans. Ind. Informat.*, vol. 17, no. 12, pp. 8003–8013, Dec. 2021, doi: 10.1109/TII.2021.3058426.

[14] Z. Geng, Z. Chen, Q. Meng, and Y. Han, "Novel transformer based on gated convolutional neural network for dynamic soft sensor modeling of industrial processes," *IEEE Trans. Ind. Informat.*, to be published, doi: 10.1109/TII.2021.3086798.

[15] X. Shi, Q. Kang, M. Zhou, J. An, and A. Abusorrah, "Novel L1 regularized extreme learning machine for soft-sensing of an industrial process," *IEEE Trans. Ind. Informat.*, vol. 18, no. 2, pp. 1009–1017, Feb. 2022.

[16] 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. Informat.*, vol. 14, no. 7, pp. 3235–3243, Jul. 2018.

[17] Q. Sun and Z. Ge, "Gated stacked target-related autoencoder: A novel deep feature extraction and layerwise ensemble method for industrial soft sensor application," *IEEE Trans. Cybern.*, to be published, doi: 10.1109/TCYB.2020.3010331.

[18] R. Chiplunkar and B. Huang, "Siamese neural network-based supervised slow feature extraction for soft sensor application," *IEEE Trans. Ind. Electron.*, vol. 68, no. 9, pp. 8953–8962, Sep. 2021.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1026–1034.

[20] S. Aranganayagi and K. Thangavel, "Clustering categorical data using silhouette coefficient as a relocating measure," in *Proc. Int. Conf. Comput. Intell. Multimedia Appl.*, 2007, pp. 13–17.

[21] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, 2003.

[22] K. Grabczewski and N. Jankowski, "Feature selection with decision tree criterion," in *Proc. 5th Int. Conf. Hybrid Intell. Syst.*, 2005, pp. 212–217.

[23] A. Altmann, L. Toloşi, O. Sander, and T. Lengauer, "Permutation importance: A corrected feature importance measure," *Bioinformatics*, vol. 26, no. 10, pp. 1340–1347, 2010.

[24] G. Ke *et al.*, "LightGBM: A highly efficient gradient boosting decision tree," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, vol. pp. 3146–3154.

[25] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[26] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural Comput.*, vol. 10, no. 7, pp. 1895–1923, 1998.

Lei Ren (Member, IEEE) received the Ph.D. degree in computer science from the Institute of Software, Chinese Academy of Sciences, Beijing, China, in 2009.

He is currently a Professor with the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China, His research interests include industrial Internet of Things, industrial big data and AI and cloud manufacturing.

Dr. Ren is an Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, *SIMULATION: Transactions of the Society for Modeling and Simulation International*, and other international journals.

Tao Wang received the B.S. degree in automation engineering in 2020 from the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China, where he is currently working toward the postgraduate degree.

His research interests include industrial intelligence, natural language processing, and soft sensors.

Yuanjun Laili (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in control science and engineering from the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China, in 2009, 2012 and 2015, respectively.

She is currently an Associate Professor with the School of Automation Science and Electrical Engineering, Beihang University. Her research interests mainly include intelligent optimization, modeling, and simulation of manufacturing systems.

Dr. Laili is an Associate Editor for the *International Journal of Modeling, Simulation, and Scientific Computing* and *Cogent Engineering*. She is a Member of the Society For Modeling and Simulation International (SCS).

Lin Zhang (Senior Member, IEEE) received the B.S. degree in control theory from Nankai University, Tianjin, China, in 1986, and the M.S. and Ph.D. degrees in control science and engineering from Tsinghua University, Beijing, China, in 1989 and 1992, respectively.

He is currently a Professor with Beihang University, Beijing. He has authored and coauthored 200 papers and 18 books and chapters. His research interests include service-oriented modeling and simulation, model engineering, and cloud manufacturing and simulation and their applications in health, etc.

Dr. Zhang was the President of the Society for Modeling and Simulation International (SCS) (2015–2016). He is a Fellow of SCS and ASIASIM, the Executive Vice President of China Simulation Federation (CSF).