Long Sequence Multivariate Time-Series Forecasting for Industrial Processes Using SASGNN

Yulong Wang, Xiaoli Wang, Jiayi Zhou, Chunhua Yang, *Fellow, IEEE*, and Yang Yang

***Abstract*—In process industries, the processes are usually very long, which results in long residence of the material in the process. In addition, in many processes, the quality of the final product cannot be detected online, making it challenging to achieve timely and effective control. Therefore, in many processes, middle-state variables are observed and used for control. In many cases, these middle-state variables cannot be measured online. Long sequence multivariate time series forecasting (MTSF) reveals the changes in the process in advance and allows us to make timely adjustments in predictive control mode. However, many current MTSF methods do not consider the relationships between variables adequately, making it difficult to achieve satisfactory results for long sequence forecasting. To address this problem, a novel sparse attention spectral graph neural network (SASGNN) is proposed. In SASGNN, a graph structure learning module is first developed to establish relationships between different variables by combining prior knowledge with a sparse attention mechanism. Then, a long sequence forecasting module transforms the constructed graph into the spectral domain to extract the latent temporal pattern of each variable and the correlation feature between different variables. Besides, considering the temporal information loss during modeling, SASGNN uses a variant GRU to extract the long-term dependencies from the historical data. Experimental results show the superior and stable performance for long sequence forecasting using the SASGNN.**

***Index Terms*—Froth flotation, graph neural network (GNN), industrial process, multivariate time-series forecasting (MTSF), sparse attention.**

I. Introduction

PROCESS industries usually consist of several substages, and each stage requires time to run, resulting in a long residence time for materials in the process. Meanwhile, many key process variables need substantial time to respond to adjustments due to physical or chemical reactions. In addition, many product indexes are difficult to detect in real time. Therefore, the control is always delayed. This is a common challenge in many process industries, which seriously affects the timeliness and effectiveness of process control.

For instance, froth flotation is a widely used separation method in mineral processing [1]. Valuable minerals are adsorbed on the surface of the froth and separated from the cell by overflow. Concentrate grade is an important product index in froth flotation. Existing researches typically predicts the final concentrate grade to provide feedback for process control [2]. However, the flotation process is dynamic and long. So, it takes a long time for the concentrate grade to respond to the adjustment of control variables, i.e., a large time delay exists in the process. In comparison, the state variables (e.g., froth image features, froth grade, and other sensor variables) in the middle of the process can respond to the adjustments with a minor time lag. Consequently, operators often use these state variables in practice to adjust the control variables. Hence, if multivariates (e.g., the froth grade and other state variables) can be accurately forecasted in advance, the flotation process can be controlled in a more timely and optimal manner, which is the same as the principle in model predictive control in many other process industries.

Furthermore, multivariate time-series forecasting (MTSF) is also useful for processes that have a too complex mechanism to build a mechanical model and have a number of important variables that cannot be measured. MTSF can model and simulate the dynamics of key variables within the process and the internal behavior of the process. This capability can enhance our understanding of the complex process and is significant for developing digital twin systems and system identification [3], especially for complex industrial processes where the first principle model cannot be established. Thus, in this article, we aim to achieve effective long sequence MTSF for more optimal and timely control in process industries and to use it as a key tool for modeling complex processes and driving the transformation of process industries to intelligent manufacturing [4].

The MTSF in the industry field has been wildly researched [5]. For instance, Liu et al. [6] discussed the problem of uneven distribution of variables and dynamic changes in industrial

Manuscript received 12 February 2024; revised 6 June 2024; accepted 27 June 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62073342, in part by the Major Program of National Natural Science Foundation of China under Grant 62394340, and in part by the High Performance Computing Center of Central South University. Paper no. TII-24-0687. *(Corresponding author: Xiaoli Wang.)*

The authors are with the School of Automation, Central South University, Changsha 410083, China (e-mail: lvguima@csu.edu.cn; xlwang@csu.edu.cn; jiayizhou@csu.edu.cn; ychh@csu.edu.cn; y_Yang2023@csu.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TII.2024.3424214.

Digital Object Identifier 10.1109/TII.2024.3424214

processes. Then, a Gaussian mixture model was used to describe the distribution of different variables, followed by an optimized particle swarm algorithm to estimate the parameters of the Gaussian regression model. Finally, a Bayesian inference strategy was applied to realize MTSF. Jiménez et al. [7] proposed a hierarchical processing strategy to improve the accuracy of MTSF, which categorized the data into simple and complex types and assigned different task models to different data types. This strategy increased the computational speed and reduced the overall model's convergence time, thus improving its efficiency.

With the development of deep learning, many deep learning-based MTSF methods have been used to improve prediction accuracy further. For example, long short-term memory (LSTM) networks have been widely used because they can effectively deal with the temporal information of variables [8]. Zhou et al. [9] developed a novel difference-LSTM to capture the influence of the differential information of secondary variables on key variables. The difference-LSTM was then used to predict the particle size of the overflow in a grinding classification process. Zhai et al. [10] used XGBoost to extract hidden features of industrial process variables. They then utilized a gated recurrent unit (GRU) to recognize temporal relationships between variables. Finally, a weighted combination of different features was used to make predictions. However, these methods mainly focus on extracting temporal information from variables, and the relationships between variables are usually ignored. Variables in the industrial field often have complex interactions that significantly affect the model's accuracy.

Graph neural networks (GNNs) can treat variables as different nodes in the graph and use edges to characterize the relationships between variables [11]. Recently, some studies have taken advantage of the spatial representation ability of GNN to realize MTSF in the industrial field. For example, Tootooni et al. [12] used graphs to represent the industrial sensor variables. They studied features and correlations between different state variables in the graph spectral domain, ultimately identifying the unusual variable fluctuations in industrial processes. Chen et al. [13] predecomposed industrial time-series data to capture latent patterns in the data at different scales. Meanwhile, a structure discovery module based on graph convolution was developed. The prediction of variables was achieved by multiscale feature fusion. Since the GNN-based model requires an accurate graph structure to obtain effective MTSF results, Granger causality (GC) is used to construct the node relationships [14]. However, the relationships between industrial variables are complicated, and methods such as GC are better suited to analyzing pairs of variables within a multivariate set, which sometimes fail to reflect the real causal relationship [15]. Therefore, current MTSF research in the industrial field faces the following challenges.

1) Many methods typically assume that the input variables are independent or do not consider the relationships between the input variables adequately. In addition, although some methods based on GNN have been used to establish relationships between variables, they neglect the prior knowledge between variables that can help to construct a more accurate graph structure.
2) Long sequence forecasting of variables in process industries is rarely mentioned in previous researches. They usually focus on predicting the final product index. However, for a long process, adjustment based on the index will cause a large delay, and accurate prediction of product index is also difficult. Therefore, the prediction of middle-state variables is significant because it can provide more timely control for the process.

To address the above-mentioned challenges, in this article, a sparse attention spectral graph neural network (SASGNN) is proposed to realize effective long sequence MTSF for industrial processes by taking the tungsten flotation process as an example. SASGNN first employs a graph structure learning module to fully establish the relationships between variables. This module utilizes the prior knowledge of variables to construct a prior adjacency matrix. In addition, it uses the sparse attention mechanism (SAM) to generate another adjacency matrix that represents the latent relationships between variables, and the final graph representation is the combination of these two matrices. Second, a long sequence forecasting module is developed for variable feature extraction. Temporal convolution networks (TCNs) extract the different scale-changing patterns of each variable, and graph convolution network (GCN) extracts the correlations between variables. Furthermore, a temporal extraction block consists of variant GRU is designed to capture the long-term temporal pattern of variables, and its output is combined with the GCN output to achieve long sequence MTSF. Comparison experiments with other MTSF models verified the effectiveness of the SASGNN.

SASGNN enables accurate long sequence prediction of key product index and middle-state variables. This capability not only facilitates the development of more effective control strategies to overcome the control delay problem but also plays a crucial role in developing digital twins in the process industries. The main contributions of this article are as follows.

1) This study proposes a novel graph structure learning method that employs SAM instead of the traditional attention mechanism to reduce computational complexity and learns the latent relationships between variables in a data-driven manner. In addition, the method incorporates prior knowledge to enhance the accuracy of the graph structure. This method also applies to other complex industrial processes and offers more advantages than traditional graph structure learning methods.
2) The proposed long sequence forecasting module in SASGNN achieves effective feature representation learning. It extracts the latent changing patterns of individual variables, the dependency relationships between variables, and captures temporal information from the long-term dynamic process. Consequently, SASGNN achieves lower error accumulation when the forecast length is long.

The rest of this article is organized as follows. Section II describes the tungsten flotation process and the formulated problem. Section III presents the detailed description of the proposed SASGNN. Section IV validates the effectiveness of the proposed method through a case study. Finally, Section V concludes this article.

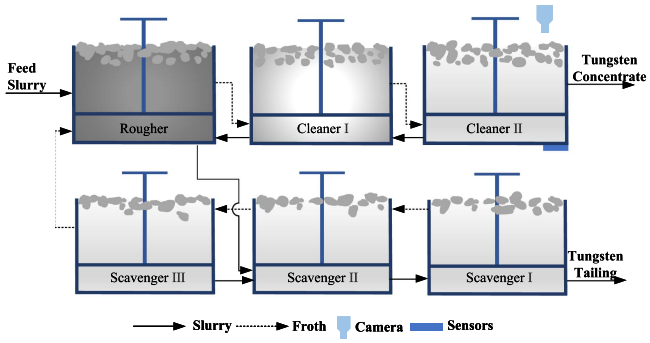

Fig. 1. Flowchart of a tungsten flotation process.

II. Process Description and Problem Formulation

Tungsten is highly significant because it is a critical material in the manufacturing industry. Tungsten flotation, a typically complex process industry, is researched as an example in this article. The primary objective of tungsten flotation is to extract tungsten concentrate with the desired grade from ores containing tungsten and maximize the recovery of tungsten. The recovery is calculated based on the concentrate grade, the feed grade, and the throughput. Thus, concentrate grade is the direct index for control. Fig. 1 illustrates the tungsten flotation process, which mainly consists of a one-stage rougher process (rougher I), two-stage cleaner processes (cleaner I and II), and three-stage scavenger processes (scavenger I, II, and III). Cleaner II is a critical step in the tungsten flotation process, as it directly affects the quality of the concentrate.

To obtain high-quality concentrate, key control variables such as reagent additions, air flow rate, and flow rate of the discharge slurry need to be adjusted timely as conditions change. However, the concentrate grade is analyzed offline every 8 h, so adjustments based on the final grade will result in a severe control delay. In contrast, the middle state variables (surface features of the froth, froth depth, etc.) are closely related to the concentrate grade. More specifically, abnormal froth depth will cause the froth to overflow too quickly or too slowly from the cell, leading to underflotation or overflotation. It is crucial to predict the froth depth and promptly adjust the air flow rate or the discharge slurry flow rate to ensure concentrate quality [16]. Furthermore, the static froth features (froth density, froth carrying rate, and froth size percentage) primarily reflect the amount of minerals carried by the froth. The color features (grayscale value, R, G, and B channel values) of the froth are closely related to the mineral type and the degree of mineralization. Froth speed is closely related to the froth depth and affects the flotation rate. Froth stability reflects the collapse and merge state of the froth.

Meanwhile, these features are much closer to the reagent addition position in the overall process and respond to adjustments much faster than the final grade. Thus, operators evaluate flotation performance through surface features of the froth and the froth depth in each flotation cell to manually adjust control variables every hour. This is the common practice in most flotation plants in China. Therefore, predicting the state variables is significant in achieving timely adjustments. Froth grade in each flotation cell is a direct variable related to the final grade; however, it cannot be measured online. Instead, froth grade is implicitly assessed by operators based on the froth's appearance and other process variables. Based on these, froth depth, froth image features, and froth grade are selected as prediction variables.

Due to the complexity of the flotation process, the interactions between these variables are complex to describe directly with an accurate mathematical model. Therefore, we use GNN to establish the relationship between them.

The GNN-based MTSF problem can be formulated as follows. Given the observed time series data of state variables $X \in \mathbb{R}^{N\times L}$, where N is the number of variables, and L is the number of input time steps, the aim is to forecast all variables in the next H time steps. Specifically, by first establishing the relationships between different variables, X is represented as graph $X^G(X, A)$, where A represents the adjacency matrix of X^G. Then, given graph sequences with length L: $X^G_{t-L+1}, \ldots, X^G_{t-1}, X^G_t$, model F is used to obtain the forecast results $\hat{Y} \in \mathbb{R}^{N\times H}$

$$\hat{Y}_{t+1}, \hat{Y}_{t+2} \ldots \hat{Y}_{t+H} = F\left(X^G_{t-L+1}, \ldots, X^G_{t-1}, X^G_t\right). \quad (1)$$

Based on the above-mentioned formulation, two main challenges need to be addressed. First, GNN-based models perform poorly if the established relationships between nodes are inaccurate. Second, long sequence forecasting usually faces the problem of error accumulation. Therefore, the method proposed in this article aims to address these two problems to achieve satisfactory forecasting results.

III. Methodology

The architecture of the proposed SASGNN is shown in Fig. 2, which consists of two modules: graph structure learning module and long sequence forecasting module.

A. Graph Structure Learning Module

The graph structure learning module aims to establish effective relationships between variables so that the raw input data can be transformed into a graph. The graph structure learning module includes an adaptive and prior learning block.

1) Adaptive Learning Block: Due to the complex mechanisms between the state variables, it is difficult to establish relationships between them directly. The attention mechanism can capture the latent relationship between different variables in a data-driven manner. However, traditional attention mechanisms often suffer from high computational complexity due to the dot product, which makes their time and memory consumption costs quadratic as L increases [17]. Thus, the adaptive learning block employs a SAM to construct the latent correlation among variables. SAM can learn the sparse feature representations of input sequences to identify the most relevant variables and reduce the computational complexity. Therefore, SAM is more suitable for long input sequences. The details of SAM are shown in Algorithm I. First, the input X is mapped into a continuous vector space by embedding, allowing the block to better capture

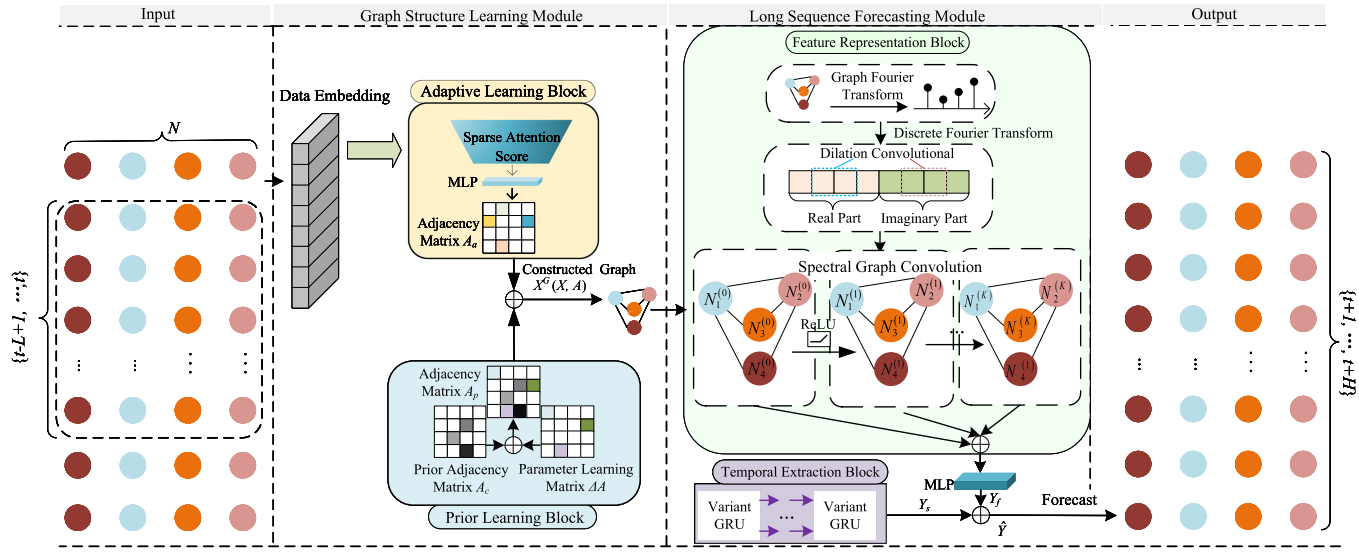

Fig. 2. Architecture of the SASGNN.

the similarities and differences between variables

$$X_v = \mathrm{FC}(X), X_p = \mathrm{PE}(X), X_e = X_v + X_p \tag{2}$$

where $X_v \in \mathbb{R}^{L \times C_e}$ is the vectorized feature representation of the variable, $X_p \in \mathbb{R}^{L \times C_e}$ is the positional representation in the sequence, and the obtained embedded vector is $X_e \in \mathbb{R}^{L \times C_e}$. FC denotes the fully connected layer, which is used to transform the dimensions of the X. PE represents the position encoding operation using sine and cosine functions. Then, different linear transform matrices W_Q and W_K are applied to X_e to generate different feature representations

$$Q = X_e W_Q, K = X_e W_K \tag{3}$$

where $Q \in \mathbb{R}^{L \times d}$, $K \in \mathbb{R}^{L \times d}$ and d is the dimension of attention heads. Next, randomly sample r times from K to obtain $\bar{K}$, and calculate the attention score $\bar{S}$ for each row element q_i in Q

$$r = L \ln L \tag{4}$$

$$\bar{S} = Q\bar{K}^T. \tag{5}$$

The score $\bar{S}$ measures how much attention q_i pays to each element $\bar{k}_i$ in $\bar{K}$. For each q_i, only part $\bar{k}_i$ has relationship with it. A sparsity measurement parameter M is defined to find the most relevant elements

$$M = \mathrm{Max}\left(\bar{S}\right) - \mathrm{Mean}\left(\bar{S}\right). \tag{6}$$

Each row element in M is obtained by calculating the difference between the maximum value and the average value of the corresponding row in $\bar{K}$. A larger M means that q_i is associated with more $\bar{k}_i$. Then, select the q_i with higher M to form $\bar{Q}$, the number of the selected q_i is z. $\bar{Q}$ contains the variables that have the most important impact on the final correlation result

$$z = 2 \ln L. \tag{7}$$

Next, using K^T to dot product with $\bar{Q}$ rather than Q, which reduces the number of dot product pairs to be calculated, as shown in Fig. 3. Compared to traditional attention mechanisms, SAM reduces the time and space complexity from $O(L^2)$ to $O(L\ln L)$.

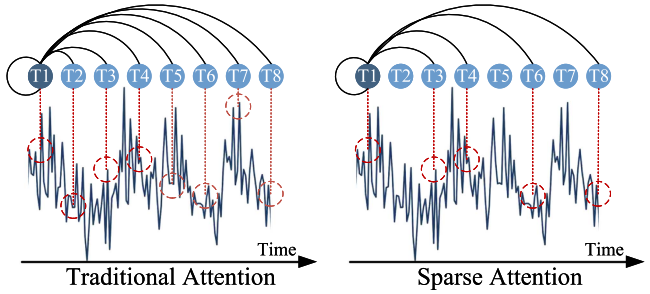

Fig. 3. Difference between SAM and traditional attention mechanism.

Algorithm 1: Sparse Attention Mechanism.

Input: Embedded vector $X_e \in \mathbb{R}^{L \times C_e}$;
Output: Sparse attention score S_a;
Start:
1: Map X_e to obtain $Q = X_e W_Q, K = X_e W_K$.
2: Random sample r times from K to obtain $\bar{K}$, $r = LlnL$.
3: Calculate the score $\bar{S} = Q\bar{K}^T$.
4: Calculate the parameter M,
$M = Max(\bar{S}) - Mean(\bar{S})$.
5: Select the q_i with higher M as $\bar{Q}$, the number of q_i is $z = 2 \ln L$.
6: Calculate the $S_a = Softmax(\bar{Q}K^T/\sqrt{d})$.
End.

Then, the softmax function is applied to obtain the sparse attention scores, which represent the degree of correlation between different variables. Finally, a multilayer perceptron (MLP)

is used to increase the flexibility of the block

$$S_a = \text{Softmax}\left(\frac{\bar{Q}K^T}{\sqrt{d}}\right) \tag{8}$$

$$A_a = \text{MLP}\left(S_a\right). \tag{9}$$

The obtained matrix $A_a \in \mathbb{R}^{N\times N}$ represents the latent relationship between different variables, which is used as part of the adjacency matrix of the final graph. Notably, due to the lower computational complexity of SAM, a multihead SAM is used to enhance the feature learning ability of the adaptive learning block. The adaptive learning block takes advantage of SAM to automatically learn the relationship between variables in a data-driven manner, which is also helpful for constructing the relationship between variables in other complex industries.

2) Prior Learning Block: Besides learning the relationships between variables solely from the data, incorporating prior knowledge about the relationships between variables can help the module better construct the adjacency matrix. For instance, the calculation of the grayscale value in color features is based on R, G, and B values. Prior knowledge can provide valuable guidance for establishing relationships between variables and increasing the interpretability of the graph structure. Hence, a learnable prior adjacency matrix A_p is defined

$$A_p = A_c + \Delta A \tag{10}$$

$$A_c = \begin{bmatrix} a_c^{11} & \cdots & a_c^{1N} \\ \vdots & a_c^{ij} & \vdots \\ a_c^{N1} & \cdots & a_c^{NN} \end{bmatrix}, a_c^{ij} = \begin{cases} 1, i \text{ and } j \text{ is related} \\ 0, \text{else} \end{cases} \tag{11}$$

$$\Delta A = \text{Softmax}\left(\text{RELU}\left(E_1 E_2^T\right)\right) \tag{12}$$

where $A_c \in \mathbb{R}^{N\times N}$ represents the adjacency matrix predefined by prior knowledge. Specifically, the grayscale value, R, G, and B values are related. The froth speed, froth depth, and froth stability are related. The a_c between these variables is set to 1. This makes it easier for the model to establish the edge between nodes where prior knowledge exists. Consequently, the final graph structure becomes more interpretable because the edge weights between these nodes are generally larger. Since A_c is fixed and lack of flexibility, a learnable parameter matrix $\Delta A \in \mathbb{R}^{N\times N}$ is used to improve the generalization ability of the block, $E_1, E_2 \in \mathbb{R}^{N\times h}$, and h represents the hidden dimension of the node features.

By combining the prior-based adjacency matrix A_p with A_a, the final graph adjacency matrix A of variables is obtained. Thus, the raw input data X can be represented as graph $X^G(X, A)$

$$A = A_a + A_p. \tag{13}$$

B. Long Sequence Forecasting Module

To achieve accurate long sequence forecasting results for state variables, the long sequence forecasting module aims to perform effective feature representation learning from the graph data X^G. This module consists of a feature representation block and a temporal extraction block.

1) Feature Representation Block: The flotation process is complex and dynamic; each variable's changes are related to its properties and correlated with other variables. This complexity makes it particularly important to learn effective feature representations of the variables that can capture both intratime-series patterns and intertime-series correlations [18]. Specifically, intratime-series patterns provide information about how a single variable changes over time, revealing its latent trends and temporal patterns. Intertime-series correlations, on the other hand, provide information about the dependencies and interactions between different variables. Therefore, the feature representation block aims to comprehensively learn the feature representations of different variables.

First, for the graph $X^G(X, A)$ obtained from the graph structure learning module, the normalized graph Laplacian matrix L^{sys} is computed

$$\widetilde{A} = I + A \tag{14}$$

$$\widetilde{A}_{\text{sym}} = \widetilde{D}^{-(1/2)}\widetilde{A}\widetilde{D}^{-(1/2)} \tag{15}$$

$$L^{\text{sys}} = I - \widetilde{A}_{\text{sym}} \tag{16}$$

where I denote the identity matrix of the same size as A, and D is the diagonal degree matrix of X^G. Next, an eigendecomposition is performed on L^{sys} to obtain the eigenvector matrix U, and U^T is used as a set of basis vectors to perform the graph Fourier transformation (GFT)

$$L^{\text{sys}} = U\Lambda U^T \tag{17}$$

$$F_g = U^T X^G. \tag{18}$$

F_g is the results of X^G after GFT. Then, rather than performing graph convolution directly, the discrete Fourier transform (DFT) is first applied to convert F_g into the frequency domain

$$X_D = \text{DFT}\left(F_g\right) \tag{19}$$

where X_D represents the frequency component. Analyzing graph signals in the frequency domain aims to capture the local patterns of each variable. In this block, the TCN employed to capture the local temporal patterns of each variable at different time scales [19]. TCN utilize dilated convolution at different layers to expand the receptive field and increase the distance over which information is propagated. This multiscale information extraction capability makes TCN more sensitive to local dependencies in sequence data, the process of TCN is denoted as follows:

$$X_T = \sum_{i=0}^{s-1} f(i) \cdot X_D\left[k - b \times i\right], k = 0, \ldots, L-1 \tag{20}$$

where $f(i)$ is the dilation convolution kernel, s is the filter size, and b is the dilation factor. Since the result of the DFT contains complex numbers, the TCN is employed separately on the real and imaginary parts. Then, X_T is transformed back to the spectral domain by the inverse discrete Fourier transform (IDFT), and the result is denoted as X_O

$$X_O = \text{IDFT}\left(X_T\right). \tag{21}$$

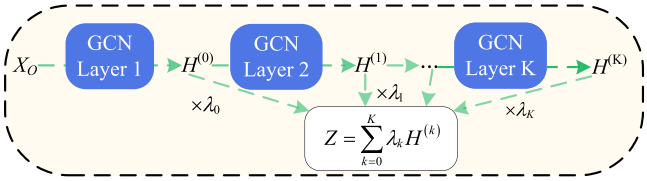

Fig. 4. Combination of the result from different GCN layers.

The X_O contains the multiscale temporal patterns of each different variable. Then, GCN is used to learn the spatial feature of the graph, i.e., the relationships between different variables. The Cth order Chebyshev polynomial T_c is used as the convolution kernel in this block

$$T_c(x) = 2xT_{c-1}(x) - T_{c-2}(x), T_0 = 1, T_1 = x. \tag{22}$$

The spatial feature information H^k learned by the kth layer of GCN can be denoted as follows:

$$H^k = \text{ReLU}\left(\sum_{j=0}^{C} \theta_j T_j(L^{\text{sys}}) H^{k-1}\right) \tag{23}$$

where θ_j represents the learnable parameters. However, GCN suffers from oversmoothing problem, i.e., as the information from neighbor nodes in different GCN layers is merged, the node features tend to converge to a fixed value, resulting in an inflexible graph structure. To overcome this problem, inspired by PageRank GCNs [20], we made a weighted combination of the results from the different GCN layers, as shown in Fig. 4.

By weighting and summing the results of the different GCN layers, the structural information of each GCN layer is preserved, and its importance is measured by the trainable parameter λ_k. Finally, Z is the output of all the GCN layers, which captures both the features of variables and the relationships between them

$$Z = \sum_{k=0}^{K} \lambda_k H^{(k)}. \tag{24}$$

2) Temporal Extraction Block: In industrial processes, temporal characteristics such as trend, periodicity, and seasonality of variables play a crucial role in making long sequence forecasts [21]. For variables such as large size froth percentage or froth density, it is hard to describe their temporal characteristics mechanistically, therefore, we have to learn their characteristics from the historical time-series data. However, it is difficult for GNN to capture such long-term temporal information because GNN is better suited to deal with spatial features. GRU is widely used to process historical information but suffer from information loss due to the gradient vanishing problem when processing long input sequences. Therefore, this block employs a variant GRU to capture the temporal patterns of variables. The updating process of the variant GRU can be described as follows:

$$z_t = \sigma\left(W_z x_t + U_z h_{t-p} + b_z\right) \tag{25}$$

$$r_t = \sigma\left(W_r x_t + U_r h_{t-p} + b_r\right) \tag{26}$$

$$c_t = \tanh\left(W_c x_t + r_t \odot U_c h_{t-p} + b_c\right) \tag{27}$$

$$h_t = (1 - z_t) \odot h_{t-p} + z_t \odot c_t \tag{28}$$

where σ represents the sigmoid function, $\odot$ denotes Hadamard product, W_z, W_r, W_c, U_z, U_r, and U_c are weights matrix. b_z, b_r, and b_c are bias, p denotes the number of hidden states to skip. In the standard GRU model, the update of the hidden state h_t considers the previous one timestep hidden state, which is h_{t-1}. However, in the variant GRU, the update of the h_t considers the hidden state from p timesteps earlier, which is h_{t-p}. The value of p is a hyperparameter to be tuned. By considering earlier hidden states, longer dependencies, and temporal characteristics in the data can be captured.

Let Y_s denote the prediction result of the temporal extraction block. Simultaneously, considering the output Z from the feature representation block, MLP is applied to generate another prediction result Y_f. Notably, we also use residual connections to improve the model performance. The final prediction of the long sequence forecasting module is obtained by combine the Y_s with Y_f. In addition, SASGNN uses mean squared error (MSE) as the loss function, where $\hat{Y}_i$ represents the predicted value and Y_i is the real value

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{Y}_i - Y_i\right)^2. \tag{29}$$

C. Complexity Analysis

In the graph structure learning module, the main time complexity comes from SAM. For the input sequence X, the time complexity of SAM is $O(L\ln L)$. In the long sequence forecasting module, the main complexity stems from the computation process of L^{sys} and graph convolution. The time complexity of (15) is $O(N^3)$. Since we use the Cth order Chebyshev polynomial as convolution kernel, the time complexity of (23) is $O(C|E|)$, where E is the number of edges in the graph. For DFT, TCN, and the variant GRU, the complexity is $O(N\ln N)$, $O(sL)$, and $O(L_{d\text{hid}})$, respectively, where d_{hid} is the dimension of the hidden state. Overall, the main complexity of SASGNN is $O(N^3)$.

IV. Case Study

In this section, the effectiveness of the proposed SASGNN is evaluated in a tungsten flotation plant and the superiority of SASGNN is demonstrated by comparing it with other popular deep learning-based MTSF methods.

A. Dataset Preparation and Evaluation

To comprehensively evaluate the effectiveness of SASGNN, we conduct experiments on the flotation dataset and two additional widely used real-world datasets. The flotation dataset was collected from the flotation plant shown in Fig. 1. All 11 variables discussed in Section II were sampled from cleaner II at the interval of 30-min over four months, from 10 May 2022 to 15 August 2022, yielding 5335 samples. The Electricity Transformer Temperature-minutely (ETTm2) dataset records data over two years for seven metrics, including voltage load, oil temperature, etc. The traffic data includes hourly road occupancy

TABLE I
BASELINE MODELS

Model type	Model name	Description
CNN/RNN	DSANet [23]	An RNN-based model combined with a dual self-attention.
	MLCNN [24]	A multi-layer CNN model combined with construal level theory of psychology.
GNN	StemGNN [18]	A GNN model based on spectral graph convolution.
	MTGNN [25]	A GNN model based on spatial graph convolution.
	GraphWaveNet [26]	A GNN model combined temporal convolution with spatial graph convolution.
Transformer	Reformer [27]	A transformer-based model using locality sensitive hashing method.
	Autoformer [22]	A transformer-based model using auto correlation decomposition.

rates measured by 862 sensors on San Francisco Bay area freeways over two years [22]. These three datasets were divided into a training set, a validation set, and a test set of 70%, 10%, and 20%, respectively.

The evaluation indices employed were the mean absolute error (MAE) and root-mean-square error (RMSE) to assess the model's performance. As the model output consists of predictions for different variables, the averages of the same indices for different variables were used as the final evaluation indices

$$\mathrm{MAE} = \sum_{o=1}^{11}\left(\frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i^o - y_i^o|\right) \tag{30}$$

$$\mathrm{RMSE} = \sum_{o=1}^{11}\left(\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i^o - y_i^o)^2}\right) \tag{31}$$

where n represents the number of sample points in the test set, $\hat{y}_i^o$ represents the predicted value of the oth variable, and y_i^o represents the true value of the oth variable.

B. Compared Models and Experiments Setting

We conduct extensive experiments with different baseline methods, which can be classified as CNN/RNN-based, GNN-based, and transformer-based methods, all of which have shown high performance in the MTSF filed. The baseline models are listed in Table I.

The experiments involved the setting of key hyperparameters based on different datasets. For the flotation dataset, the input window size L was selected from 2 to 40, with intervals of 2; the number of attention heads N_A was selected from {4, 8, 16}; the number of variant GRU neurons N_g was chosen from 50 to 400, with intervals of 50; the skip interval p was selected from 2 to 40, with intervals of 2; the dilation factor b in TCN was selected from {1, 2, 4}; the TCN kernel size s was selected from {2, 4, 6}; the number of TCN neurons N_c was selected from 50 to 400, with intervals of 50; the number of GCN network layers N_s was selected from{2, 3, 4}; and the mini-batch size B_n was selected from {16, 32, 64}. In addition, to compare the long sequence forecasting ability of different models, it is most reasonable to predict the operating conditions within the next four hours based on the nature of the flotation process; therefore, H is set to {1,3,5,7,9}.

For the ETTm2 and the traffic dataset, we followed the experimental settings of Autoformer in Wu et al.'s [22] work, where the input length was fixed to 96, and the horizon lengths are fixed to {96, 192, 336}. The minibatch size B_n was selected from {64, 128, 256, 512}, and the TCN kernel size s was selected from {4, 6, 8, 10}. Other hyperparameter ranges were chosen in the same range as for the flotation dataset.

The optimal hyperparameters settings for different models were selected using the grid search method from the validation set. All three datasets were preprocessed through standardization, and according to the efficiency of Adam optimizer [30], it was used to solve the optimization problem in this article. An early stop strategy was set to prevent underfitting or overfitting. Following Wu et al.'s [22] work, the evaluation indices for the all datasets were calculated using normalized forecasting results for better comparison. The entire experiment was conducted on the platform with Intel i7-12700H, 8.0 GB RAM, and NVIDIA RTX 3060, programmed in Python 3.8 and implemented using the PyTorch framework. The code is available at.[1]

C. Results and Discussions

The evaluation indices of all models on different datasets are presented in Tables II–IV. It can be seen that SASGNN achieves high performance in all three datasets, and Fig. 5 further shows the prediction results by different methods for the froth grade of the flotation dataset when H is 3.

Compared with the CNN/RNN-based model, it can be seen from Fig. 5(a), (b), and (h) that although DSANet and MLCNN can predict the trend of the variables, they are not sensitive to the fluctuation of the local conditions, which makes them unable to provide reliable information about short-term working conditions when the operation needs to be adjusted in real time. In contrast, SASGNN provides accurate forecasts of both short-term fluctuations and long-term trends, as it captures more useful information than CNN/RNN-based models by exploiting the correlation between variables and temporal characteristics.

From Fig. 5(c), (d), (e), and (h), it can be seen that the GNN-based model are able to better predict the true fluctuation values of the grade change, mainly because they effectively exploit the strong interconnections between different variables in the flotation process, enabling them to obtain more local variable information than the CNN/RNN-based models. GraphWaveNet also builds relationships between variables in a data-driven manner, but it does not take advantage of the prior knowledge between the variables. Although it predicts fluctuation points better than MTGNN and StemGNN, it still has a large error in its prediction results compared to SASGNN.

From Fig. 5(f)–(h), and Table II, it can be seen that the transformer-based model can model long-range dependencies

[1][Online]. Available: github.com/ywmcsu/SASGNN

TABLE II
EVALUATION INDICES OF DIFFERENT MODELS ON FLOTATION DATASET

Model	H=1		H=3		H=5		H=7		H=9	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
DSANet	0.3809	0.6035	0.4199	0.6479	0.4667	0.7046	0.4497	0.6988	0.4889	0.7581
MLCNN	0.3911	0.6105	0.4255	0.6518	0.4511	0.7136	0.4764	0.7297	0.4960	0.7593
StemGNN	0.4340	0.6492	0.4608	0.6825	0.4943	0.7306	0.5215	0.7555	0.5363	0.7913
MTGNN	0.3701	0.5822	0.4218	0.6452	0.4487	0.6889	0.5002	0.7290	0.4971	0.7303
GraphWaveNet	0.3890	0.5967	0.3924	0.6276	0.4497	0.6921	0.4567	0.7187	0.4988	0.7671
Reformer	0.3885	0.5993	0.4170	0.6392	0.4528	0.6776	0.4551	0.7168	0.4755	0.7214
Autoformer	0.3682	0.6012	0.3920	0.6233	**0.4119**	0.6540	**0.4414**	0.7096	0.4682	0.7208
SASGNN	**0.3560**	**0.5733**	**0.3893**	**0.6038**	0.4262	**0.6514**	0.4556	**0.6906**	**0.4634**	**0.7195**

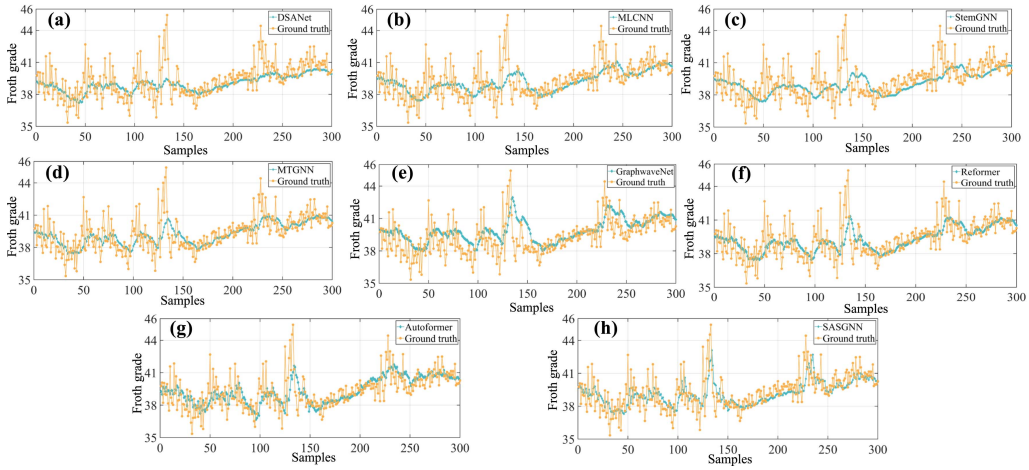

Fig. 5. Prediction results of froth grade by different models. (a) DSANet. (b) MLCNN. (c) StemGNN. (d) MTGNN. (e) GraphwaveNet. (f) Reformer. (g) Autoformer. (h) SASGNN.

TABLE III
EVALUATION INDICES OF DIFFERENT MODELS ON ETTM2 DATASET

Model	Horizon(96/192/336)	
	MAE	RMSE
DSANet	1.3482/1.3793/1.4246	1.7438/1.7527/1.7621
MLCNN	0.5341/0.6228/0.8152	0.6596/0.8544/1.0959
StemGNN	0.4498/0.5712/0.8742	0.6042/0.7301/1.1675
MTGNN	0.3348/0.4347/0.7542	0.4930/0.6261/0.9654
GraphWaveNet	0.8264/1.1454/1.2148	1.0242/1.4314/1.5159
Reformer	0.6192/0.8267/0.9723	0.8113/1.0382/1.2444
Autoformer	0.3392/**0.3401**/0.3716	0.5048/**0.5297**/0.5823
SASGNN	**0.2748**/0.3584/**0.3694**	**0.4930**/0.5394/**0.5608**

TABLE IV
EVALUATION INDICES OF DIFFERENT MODELS ON TRAFFIC DATASET

Model	Horizon (96/192/336)	
	MAE	RMSE
DSANet	0.7761/0.7864/0.8019	1.1992/1.2095/1.2161
MLCNN	0.5470/0.6172/0.7844	0.7887/0.8597/1.0020
StemGNN	0.4011/0.3814/0.4157	0.8542/0.8285/0.8760
MTGNN	0.4288/0.4924/0.4812	0.8933/0.9512/0.9320
GraphWaveNet	0.5945/0.6014/0.7684	0.9990/1.0043/1.0794
Reformer	0.4231/0.4201/0.4204	0.8555/0.8562/0.8614
Autoformer	0.3880/0.3824/**0.3369**	0.7828/0.7850/**0.7885**
SASGNN	**0.3727**/**0.3696**/0.3754	**0.7812**/**0.7792**/0.7901

in sequence data. By decomposing complex time-series data into smooth components, Autoformer achieves competitive prediction results; however, the predictions of the sample intervals from 100 to 150 and from 200 to 250 show that SASGNN performs better in predicting abnormal conditions. This is mainly because SASGNN effectively exploits the correlation between variables in the flotation process using graph convolution while mitigating the oversmoothing problem in the feature aggregation process.

D. Efficiency Analysis

We use flops and parameters to compare the efficiency performance of the models. Flops is a measure of computational performance that indicates how many floating-point operations can be performed per second, which can measure the complexity of the model. Parameters usually refer to the learnable weights and biases in the model and are often used as important indicators to measure the size of the model. We use the flotation dataset with a fixed L of 20 and a fixed H of 3. The results are shown in Table V. From Table V, it can be seen that CNN/RNN-based model have fewer FLOPs and parameters, giving them an advantage in running speed and memory consumption; however, their prediction results have larger errors. The GNN-based model has significantly higher FLOPs due to their frequent use of multiple matrix decompositions and multiplications. In contrast, transformer-based models generally have lower FLOPs but tend to have more parameters, making their training process more time-consuming. SASGNN has a lower number of parameters than Autoformer, as well as lower FLOPs than the GNN models. Despite the high complexity of SASGNN, the forecasting step

TABLE V
EFFICIENCY INDICES FOR DIFFERENT MODELS

Models	FLOPs	Parameters
DSANet	79,845,411	604,657
MLCNN	54,161,924	160,335
StemGNN	1,092,175,872	2,934,473
MTGNN	564,782,564	1,754,657
GraphWaveNet	447,537,421	1,001,764
Reformer	166,110,848	8,805,068
Autoformer	159,118,576	10,561,548
SASGNN	405,219,328	6,082,760

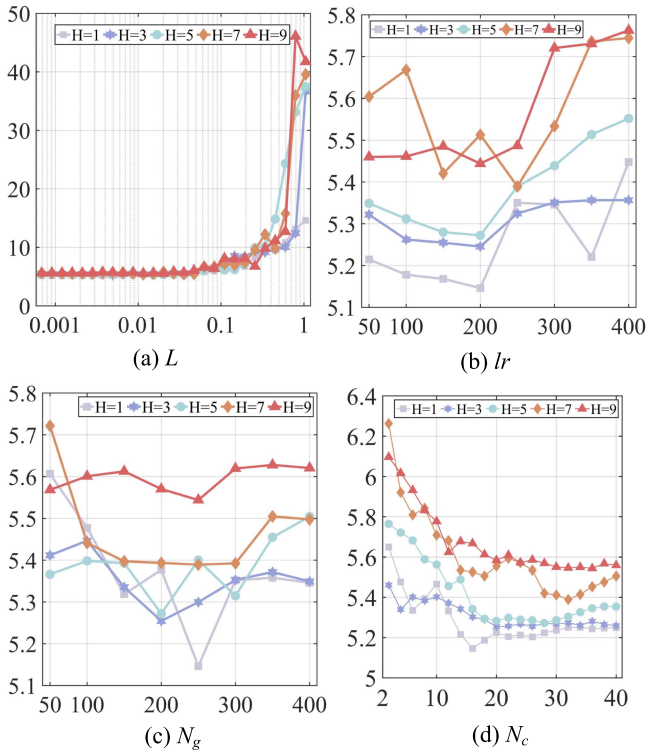

Fig. 6. Hyperparameter study results.

size in flotation plant is typically tens of minutes, making the cost of inference affordable. This makes SASGNN more valuable in practice under the premise of ensuring better prediction accuracy.

E. Hyperparameter Sensitivity Analysis

We conduct a hyperparameter sensitivity study on four key hyperparameters that influence the prediction performance of SASGNN, including the input window size L, learning rate l_r, number of GRU neurons N_g, and number of TCN neurons N_c. The experiments were conducted on the flotation dataset using different H, and Fig. 6 shows the experimental results of the hyperparameter study. As shown in the Fig. 6(a), for different L, taking $H = 3$ as an example, the RMSE initially decreases as the L increases. When it decreases to a certain level, the RMSE no longer changes significantly, indicating that, at this point, increasing the window size no longer provides additional information to the model but instead increases the computational cost. When the input length is fixed, e.g., $L = 20$, it can be seen that the larger the H is, the larger the corresponding RMSE is, which indicates that the prediction of long sequence often requires a longer input window to provide sufficient information. From Fig. 6(b) it can be seen that for larger H, the corresponding RMSE increases faster with the increasing of the learning rate compared to the smaller H. This suggests that the model is more sensitive to the choice of learning rate when predicting large H. We assume this is mainly because of the error cumulative effect when predicting larger length. When the learning rate is low (less than 0.01), the model maintains a low RMSE for all different length. From Fig. 6(c) and (d), it can be seen that when the number of neurons is small, the model does not achieve the best prediction for different step sizes due to the underfitting problem. As the number of neurons increases, the model's predictive ability starts to improve, and the RMSE starts to decrease. When the number of neurons increases to a certain extent, the RMSE increases because the model becomes overly complex, leading to overfitting and a decrease in prediction accuracy.

TABLE VI
ABLATION STUDY DESIGN DETAILS

Label	Study name	Description
1	w/o Wa	Remove the adaptive learning block of SASGNN.
2	w/o Wp	Remove the prior learning block of SASGNN.
3	w/o P	Remove the temporal extraction block.
4	w/ KNN	K-nearest neighbor (KNN) clustering is used to establish the adjacency matrix.
5	w/ CATs	Using CATs to replace the feature representation block.
6	w/ ADSF	Using ADSF to replace the feature representation block.
7	SASGNN	Complete SASGNN model structure.

F. Ablation Study

An ablation study was conducted to verify the effectiveness of the key components in SASGNN. Some core parts of SASGNN were replaced by different methods to validate its flexibility. The graph conjoint attention networks and adaptive structural fingerprint (ADSF) both achieve effective graph convolution performance by learning the higher order structure on the graph [28], [29]. This study used the flotation dataset with a fixed L of 20 and a fixed H of 3. The study details are given in Table VI, and the results are shown in Fig. 7.

Comparing the results of ablation studies (1), (2), and (7) reveals that using only the prior learning block yields lower prediction errors than using only the adaptive learning block. This demonstrates the importance of prior knowledge in providing valuable initial graph structure information to the model, leading to more accurate predictions. The best results were obtained by combining the adaptive learning block with the prior learning block. The results of the ablation studies (3) and (7) highlight the importance of capturing long-term dependencies, as latent temporal characteristics often need to be extracted from historical data. Studies (4) and (7) show that the graph structure

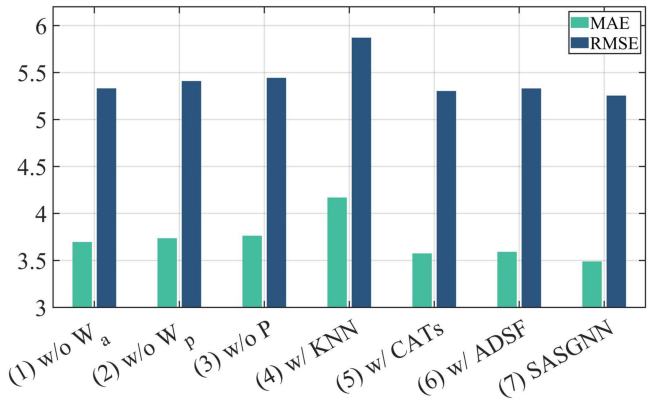

Fig. 7. Ablation study results.

established by the KNN clustering method lacks accuracy, as this distance-based measurement method does not perform well in constructing relationships between complex industrial variables. The performance of studies (5) and (6) is highly competitive, as they both focus on extracting higher order graph structure information from neighbor nodes, which can mitigate the oversmoothing problem. However, SASGNN not only extracts both intra time-series patterns and inter time-series correlations of variables, but also mitigates the oversmoothing problem by aggregating information from different GCN layers, ultimately achieving the best prediction results in the studies (7).

V. Conclusion

In this article, a novel SASGNN model is proposed to realize effective long sequence MTSF for complex process industries. The graph structure learning module in SASGNN not only extract the latent correlations between variables through SAM but also takes full advantage of prior knowledge. This combination provides a more effective graph structure. Meanwhile, the long sequence forecasting module in SASGNN achieves effective feature extraction, where TCN extracted the latent local change patterns of individual variables, and GCN captured the relationship between variables; furthermore, a variant GRU was used to improve the ability of the model to learn the temporal characteristics from variables. The case study on different datasets verifies that SASGNN outperforms other deep learning-based MTSF methods in prediction accuracy and error accumulation, suggesting that SASGNN can provide more reliable information for timely control and is more suitable for real-world industrial applications.

Future work will consider dynamic graph structures to address the working condition drift problem and explore ways to reduce the computational complexity of SASGNN as the number of nodes increases. Furthermore, future research will leverage the prediction of SASGNN results to develop optimal control strategies and explore the integration of SASGNN with digital twins to achieve comprehensive and accurate process modeling and control.

References

[1] X. Wang, C. Song, C. Yang, and Y. Xie, "Process working condition recognition based on the fusion of morphological and pixel set features of froth for froth flotation," *Minerals Eng.*, vol. 128, pp. 17–16, Nov. 2018.
[2] H. Zhang, Z. Tang, Y. Xie, Z. Yin, and W. Gui, "ES-Net: An integration model based on encoder–decoder and siamese time series difference network for grade monitoring of zinc tailings and concentrate," *IEEE Trans. Ind. Electron.*, vol. 70, no. 11, pp. 11819–11830, Nov. 2023.
[3] F. Tao, H. Zhang, A. Liu, and A. Y. C. Nee, "Digital twin in industry: State-of-the-art," *IEEE Trans. Ind. Informat.*, vol. 15, no. 4, pp. 2405–2415, Apr. 2019.
[4] A. Essien and C. Giannetti, "A deep learning model for smart manufacturing using convolutional LSTM neural network autoencoders," *IEEE Trans. Ind. Informat.*, vol. 16, no. 9, pp. 6069–6078, Sep. 2020.
[5] S. K. Sogi and S. Kumar Mittal, "A comprehensive review and analysis for forecasting industrial data," in *Proc. ICSCCC - Int. Conf. Secur. Cyber Comput. Commun.*, Jalandhar, India, 2021, pp. 155–159.
[6] T. Liu, H. Wei, S. Liu, and K. Zhang, "Industrial time series forecasting based on improved gaussian process regression," *Soft Comput.*, vol. 24, no. 20, pp. 15853–15869, Oct. 2020.
[7] M. J. Jiménez-Navarro, M. Martínez-Ballesteros, F. Martínez-lvarez, and G. Asencio-Cortés, "PHILNet: A novel efficient approach for time series forecasting using deep learning," *Inf. Sci.*, vol. 632, pp. 815–832, Jun. 2023.
[8] X. Zhang, Y. Lei, H. Chen, L. Zhang, and Y. Zhou, "Multivariate time-series modeling for forecasting sintering temperature in rotary kilns using DCGNet," *IEEE Trans. Ind. Informat.*, vol. 17, no. 7, pp. 4635–4645, Jul. 2021.
[9] J. Zhou, X. Wang, C. Yang, and W. Xiong, "A novel soft sensor modeling approach based on difference-LSTM for complex industrial process," *IEEE Trans. Ind. Informat.*, vol. 18, no. 5, pp. 2955–2964, May 2022.
[10] N. Zhai, P. Yao, and X. Zhou, "Multivariate time series forecast in industrial process based on XGBoost and GRU," in *Proc. ITAIC-IEEE Joint Int. Inf. Technol. Artif. Intell. Conf.*, Chongqing, China, 2020, pp. 1397–1400.
[11] T. N. Kipf and M. Welling, "Semi-Supervised classification with graph convolutional networks," Feb. 2017. *arXiv:1609.02907.*
[12] M. S. Tootooni, P. K. Rao, C.-A. Chou, and Z. J. Kong, "A spectral graph theoretic approach for monitoring multivariate time series data from complex dynamical processes," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 1, pp. 127–144, Jan. 2018.
[13] L. Chen et al., "Multi-scale adaptive graph neural network for multivariate time series forecasting," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 10, pp. 10748–10761, Oct. 2023.
[14] J. Li, Y. Shi, H. Li, and B. Yang, "TC-GATN: Temporal causal graph attention networks with nonlinear paradigm for multivariate time series forecasting in industrial processes," *IEEE Trans. Ind. Informat.*, vol. 19, no. 6, pp. 7592–7601, Jun. 2023.
[15] S. Hu, X. Jia, J. Zhang, W. Kong, and Y. Cao, "Shortcomings/Limitations of blockwise Granger causality and advances of blockwise new causality," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 12, pp. 2588–2601, Dec. 2016.
[16] K. Hadler, M. Greyling, N. Plint, and J. J. Cilliers, "The effect of froth depth on air recovery and flotation performance," *Minerals Eng.*, vol. 36–38, pp. 248–253, Oct. 2012.
[17] H. Zhou et al., "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 11106–11115.
[18] D. Cao et al., "Spectral temporal graph neural network for multivariate time-series forecasting," in *Adv. Neural Inf. Proces. Syst.*, Curran Associates, Inc., 2020, pp. 17766–17778.
[19] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," Apr. 2018, *arXiv:1803.01271*.
[20] C. Eli, J. Peng, P. Li, and M. Olgica, "Adaptive universal generalized PageRank graph neural network," in *Proc. ICLR - Int. Conf. Learn. Represent.*, Virtual, Online, 2021, pp. 7562–7577.
[21] G. Lai, W. C. Chang, Y. Yang, and H. Liu, "Modeling long- and short-term temporal patterns with deep neural networks," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval.*, New York, NY, USA 2018, pp. 95–104.
[22] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," in *Proc. Adv. Neural Inf. Process. Syst.*, Virtual, Online, 2021, pp. 22419–22430.

[23] S. Huang, D. Wang, X. Wu, and A. Tang, "DSANet: Dual self-attention network for multivariate time series forecastingt," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, Beijing, China. 2019, pp. 2129–2132.
[24] J. Cheng, K. Huang, and Z. Zheng, "Towards better forecasting by fusing near and distant future visions," in *Proc. AAAI Conf. Artif. Intell.*, New York, NY, USA, 2020, pp. 3593–3600.
[25] Y. Liu et al., "Multivariate time-series forecasting with temporal polynomial graph neural networks," in *Adv. Neural Inf. Process. Syst.*, New Orleans, LA, USA, 2022, pp. 19414–19426.
[26] Z. Wu et al., "Graph wavenet for deep spatial-temporal graph modeling," in *Proc. 28th IJCAI Int. Joint Conf. Artif. Intell.*, Macao, China, 2019, pp. 1907–1913.
[27] N. Kitaev, Ł. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," in *Proc. Int. Conf. Learn. Represent.*, Addis Ababa, Ethiopia, 2020, pp. 12764–12776.
[28] T. He, Y. Ong, and L. Bai, "Learning conjoint attentions for graph neural Nets," in *Adv. Neural Inf. Process. Syst.*, Virtual Online, 2021, pp. 2641–2653.
[29] K. Zhang, Y. Zhu, J. Wang, and J. Zhang, "Adaptive structural fingerprints for graph attention networks," in *Proc. Int. Conf. Learn. Represent.*, Addis Ababa, Ethiopia, 2020, pp. 6602–6614.
[30] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 1, pp. 249–270, Jan. 2022.

Yulong Wang received the B.Eng. degree in electronic information engineering and the M.Eng. degree in mineral processing from the Taiyuan University of Technology, Taiyuan, China, in 2018 and 2022, respectively. He is currently working toward the Ph.D. degree in control science and engineering with the School of Automation, Central South University, Changsha, China.

His research interests include data modeling and control of complex industrial processes, data analysis, and machine learning.

Xiaoli Wang received the Ph.D. degree in control theory and control engineering from Central South University, Changsha, China, in 2011.

She is currently a Full Professor with the School of Automation, Central South University. From December 2016 to December 2017, she was a Visiting Scholar with the Department of Energy Institute, Texas A&M University, College Station, TX, USA. Her research interests include modeling and optimal control of the complex industrial processes, machine learning and pattern recognition, industrial process soft sensor modeling, and process data analysis.

Jiayi Zhou received the B.Eng. degree in automation and the M.Eng. degree in control science and engineering from Chongqing University, Chongqing, China, in 2014 and 2017, respectively, and the Ph.D. degree in control science and engineering from the School of Automation, Central South University, Changsha, China, in 2024.

Her research interests include modeling and optimal control of complex industrial processes and soft sensor modeling.

Chunhua Yang (Fellow, IEEE) received the M.Eng. degree in automatic control engineering and the Ph.D. degree in control science and engineering from Central South University, Changsha, China, in 1988 and 2002, respectively.

She is currently a Full Professor with Central South University. From 1999 to 2001, she was with the Department of Electrical Engineering, Katholieke Universiteit Leuven, Leuven, Belgium. Her research interests include modeling and optimal control of complex industrial processes, intelligent control systems, and fault-tolerant computing of real-time systems.

Yang Yang received the B.Eng. degree in automation from Yanshan University, Qinhuangdao, China, in 2020, and the M.Eng. degree in electronic information from Central South University, Changsha, China, in 2023. He is currently working toward the Ph.D. degree in control science and engineering with the School of Automation, Central South University.

His research interests include modeling and optimal control of complex industrial processes, intelligent control systems, and computer vision.