Systems & Control Transactions

Research Article - Peer Reviewed Conference Proceeding
ESCAPE 35 - European Symposium on Computer Aided Process Engineering
Ghent, Belgium. 6-9 July 2025
Jan F.M. Van Impe, Grégoire Léonard, Satyajeet S. Bhonsale,
Monika E. Polańska, Filip Logist (Eds.)

PSE PRESS

Data-Driven Dynamic Process Modeling Using Temporal RNN Incorporating Output Variable Autocorrelation and Stacked Autoencoder

Yujie Hu[a], Lingyu Zhu[c], Han Gong[d], Xi Chen[a,b]*

[a] State Key Laboratory of Industrial Control Technology, College of Control Science and Engineering, Zhejiang University, Hangzhou, Zhejiang, 310027, China.
[b] Huzhou Institute of Industrial Control Technology, Zhejiang, China.
[c] College of Chemical Engineering, Zhejiang University of Technology, Hangzhou, Zhejiang, 310014, China
[d] Zhejiang Amino-Chem Co., Ltd, Shaoxing, Zhejiang, 312369, China
* Corresponding Author: xi_chen@zju.edu.cn.

ABSTRACT

Dynamic process modeling in process industries has been extensively studied, especially with the development of deep learning techniques. Recurrent neural networks (RNN) and stacked autoencoders (SAE) are two powerful tools for dynamic modeling and data processing. However, most existing research primarily focuses on extracting features from process input data, often neglecting the temporal autocorrelation of output variables. In this work, a hierarchical model based on time-series RNN structure is proposed. The upper layer employs a long short-term memory (LSTM) network to extract temporal features from process input data. The lower layer uses a gated recurrent unit (GRU) to model the temporal dependencies of output variables across samples. These two parts are concatenated to form the model. Additionally, SAE is utilized to perform dimensionality reduction and reconstruction of process input, seamlessly integrating the reconstruction process with the RNN into a unified framework, termed the AR-SAE-RNN model. Distributed training is employed to effectively learn the model parameters. The proposed AR-SAE-RNN model has been applied to an industrial distillation column. The results demonstrate the model's effectiveness in capturing the dynamic behavior of the system.

Keywords: Dynamic process modeling, RNN, SAE

1. INTRODUCTION

In recent years, with the rapid advancement of production and information technologies, as well as the shift towards customized and refined product demands, process industries, represented by chemical processes, have become increasingly complex. The various coupled unit operations and energy consumption reduction strategies have made process industry modeling a challenge. On the timescale, process industry modeling can be classified into two categories: steady-state models and dynamic models, with dynamic models playing a dominant role as they better capture the temporal dependencies in real-world production processes. There are generally three approaches to dynamic process modeling: mechanistic-driven, data-driven, and hybrid mechanistic-data-driven models. Among these, mechanistic-driven and hybrid-driven models typically involve the modeling and solving of ordinary differential equations (ODE) or differential-algebraic equations (DAE), which require significant computational resources and time. In contrast, data-driven models, due to their simplicity and low dependence on process-specific backgrounds, have strong potential for application in dynamic modeling.

Data-driven models, in short, involve distinguishing between input and output variables of process data and constructing black-box models between these variables. The burgeoning amount of process data in the process industry has highlighted the advantages of deep learning techniques **[1]**. Among these, structures such as stacked autoencoders (SAE) **[2]**, recurrent neural networks (RNN) **[3]**, and their variants have been widely applied **[4,5]**.

Stacked Autoencoders and their variants are powerful tools for semi-supervised learning and handling missing data. When applied to dynamic modeling, they can perform feature extraction, dimensionality reduction, and denoising on the input data in a semi-supervised manner, making the data more suitable as model inputs **[6-8]**. RNN can extract temporal features from input variables, and variants such as gated recurrent units (GRU) and long short-term memory (LSTM) networks, by introducing gating mechanisms and memory units, further address the issues of vanishing and exploding gradients in traditional RNN, making RNN increasingly powerful in dynamic modeling **[9,10]**.

However, research on the application of RNN to dynamic modeling in process industries has primarily focused on the input process variables **[10,11]**, neglecting the autocorrelation of output variables over time. Specifically, the past states of output variables can also influence the current state. Although some studies **[12]** have included past output variables as inputs, this approach requires the actual values of the output variables from previous time steps, which imposes high demands in practical industrial settings and may introduce delays. Therefore, for dynamic systems with highly autocorrelated output variables, it is crucial to indirectly incorporate the influence of historical states into the RNN model.

Furthermore, most research on the application of SAE and its variants to dynamic modeling typically first uses SAE for input reconstruction, then constructs a dynamic model based on the reconstructed inputs, with the two processes being independent and lacking effective integration. Based on this, this paper proposes a temporal RNN model with an integrated SAE module that considers the autocorrelation of output variables. The model effectively learns the parameters through distributed training, and its effectiveness is validated in a real-world soft sensor case for a distillation column.

2. MODEL CONSTRUCTION AND TRAINING

In dynamic modeling, the conventional approach typically involves using RNN models to extract temporal features from input data for each sample, followed by fitting the corresponding output data. However, this approach often neglects the autocorrelation of the output variables or the inter-sample correlations. To address this limitation, we propose a hierarchical model. The upper layer extracts temporal features from input data within each sample, while the lower layer captures temporal dependencies of output data across samples. The features from both layers are concatenated to predict the output variables. Given the complexity of processing high-dimensional input data in the upper layer, we employ the LSTM model for its advanced feature extraction capabilities. In contrast, the lower layer focuses solely on the temporal modeling of output variables, allowing us to simplify the structure by utilizing a GRU model. Additionally, we incorporate a SAE module to reconstruct input data, effectively integrating input reconstruction and model regression into a unified framework. This integrated model is termed the Autoregressive-Stacked Autoencoder-Recurrent Neural Network (AR-SAE-RNN).

2.1 Sample Reorganization

Before introducing the AR-SAE-RNN model, we preprocess the standard sample data by reorganizing it. For an input-output dataset ($\boldsymbol{X'}$, $\boldsymbol{Y'}$):

$$X' = [x_i']_{i=1}^{N} \tag{1}$$

$$Y' = [y_i']_{i=1}^{N} \tag{2}$$

where $\boldsymbol{N}$ is the total number of samples, $\boldsymbol{x_i}$ represents the input data of the $\boldsymbol{i}$-th sample, and $\boldsymbol{y_i}$ is its corresponding output data. We then reorganize the samples, combining $\boldsymbol{K}$ consecutive original samples into a single new sample:

$$x_j = [x_i', x_{i+1}', \cdots, x_{i+K-1}']_{i=j} \tag{3}$$

$$y_j = [y_{i+K-1}']_{i=j} \tag{4}$$

To maximize data utilization, we set the sliding step size to 1 during the reorganization process, generating the following reorganized samples:

$$X = [x_j]_{j=1}^{N-K+1} \tag{5}$$

$$Y = [y_j]_{j=1}^{N-K+1} \tag{6}$$

After reorganization, the total number of samples becomes $\boldsymbol{N-K+1}$, effectively discarding the first $\boldsymbol{K-1}$ original samples. In the reorganized dataset, the $\boldsymbol{K}$ input samples are used to predict the output variables of the $\boldsymbol{K}$-th sample.

2.2 Model Architecture

The structure of the proposed AR-SAE-RNN model is shown in Fig. 1 and comprises three components:

(1) SAE Module: This module performs dimensionality reduction and reconstruction of the raw input $\boldsymbol{x}$, extracting deep features $\boldsymbol{z}$.

(2) LSTM Module: Using the deep features $\boldsymbol{z}$ extracted from the SAE module as input, this module captures temporal features of the input data within each sample, generating hidden states $\boldsymbol{h_t^k}$.

(3) GRU Module: Taking the hidden state $\boldsymbol{h_T^k}$ of the $\boldsymbol{T}$-th time step of the LSTM unit in each sample as input, this module extracts temporal features of the output variables across samples.

Finally, the hidden state $\boldsymbol{s_{k+K}}$ of the $\boldsymbol{K}$-th sample in the GRU module is concatenated with the hidden state $\boldsymbol{h_T^{k+K}}$ of the $\boldsymbol{T}$-th time step in the LSTM module to predict the output variables of the $\boldsymbol{K}$-th sample.

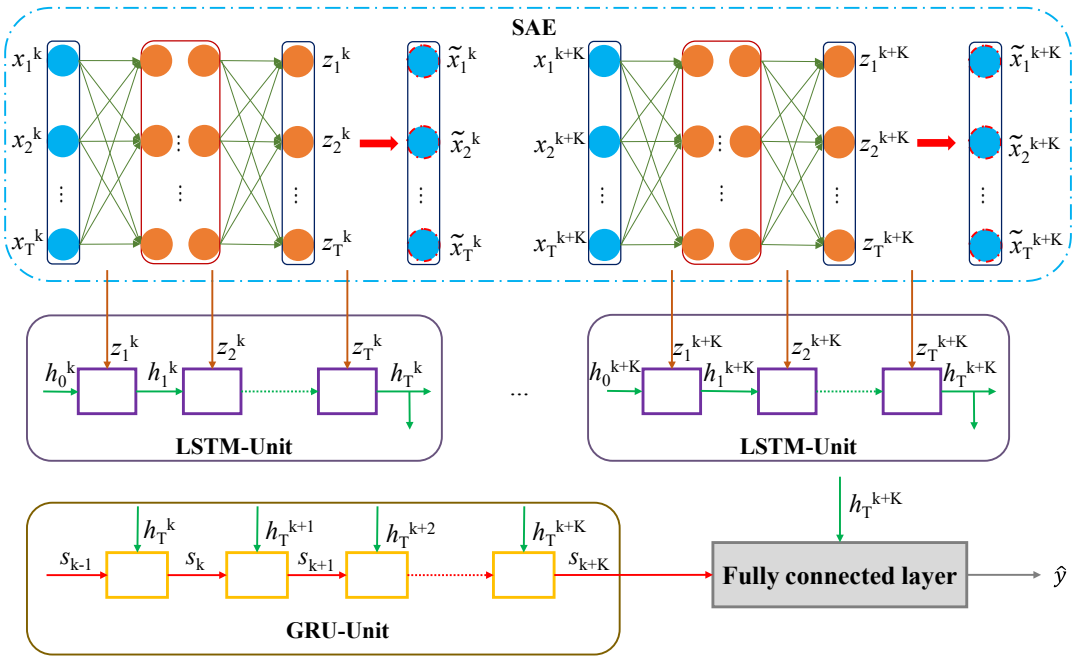

Figure 1. AR-SAE-RNN Model Architecture.

Assuming there are ***E*** stacked layers, the SAE module can be expressed as:

$$z^{(0)} = x \tag{7}$$

$$z^{(e)} = \sigma\left(W_{SAE}{}^{(e)} z^{(e-1)} + b_{SAE}{}^{(e)}\right) \tag{8}$$

$$e = 1,2,\cdots,E \tag{9}$$

where $\boldsymbol{W_{SAE}}$ and $\boldsymbol{b_{SAE}}$ are trainable parameters, $\boldsymbol{\sigma}$ is the activation function, and ***E*** is a hyperparameter.

The second component is a standard LSTM unit, except that its input is the deep feature ***z*** from the SAE module instead of the raw input ***x***. The LSTM unit model for the ***k***-th sample at the ***t***-th time step is defined as:

$$f_{(t)} = \sigma\left(W_{fz} z_t^k + W_{fh} h_{t-1}^k + b_f\right) \tag{10}$$

$$i_{(t)} = \sigma\left(W_{iz} z_t^k + W_{ih} h_{t-1}^k + b_i\right) \tag{11}$$

$$o_{(t)} = \sigma\left(W_{oz} z_t^k + W_{oh} h_{t-1}^k + b_o\right) \tag{12}$$

$$\tilde{C}_{(t)} = tanh\left(W_{cz} z_t^k + W_{ch} h_{t-1}^k + b_c\right) \tag{13}$$

$$C_{(t)} = f_{(t)} \odot C_{(t-1)} + i_{(t)} \odot \tilde{C}_{(t)} \tag{14}$$

$$h_t^k = o_{(t)} \odot tanh\left(C_{(t)}\right) \tag{15}$$

$$t = 1,2,\cdots,T \tag{16}$$

where $\boldsymbol{W_{fz}}$, $\boldsymbol{W_{fh}}$, $\boldsymbol{b_f}$, $\boldsymbol{W_{iz}}$, $\boldsymbol{W_{ih}}$, $\boldsymbol{b_i}$, $\boldsymbol{W_{oz}}$, $\boldsymbol{W_{oh}}$, $\boldsymbol{b_o}$, $\boldsymbol{W_{cz}}$, $\boldsymbol{W_{ch}}$, $\boldsymbol{b_c}$ are trainable parameters, $\boldsymbol{\sigma}$ and ***tanh*** are activation functions, and ***T*** is a hyperparameter.

The third component is a standard GRU unit, with its input being the hidden state output by the LSTM module. The GRU model for the ***k***-th sample (considered as the ***k***-th time step in the GRU module) is defined as:

$$r_{(k)} = \sigma\left(W_r h_T^k + U_r s_{k-1}\right) \tag{17}$$

$$gz_{(k)} = \sigma\left(W_{gz} h_T^k + U_{gz} s_{k-1}\right) \tag{18}$$

$$\tilde{s}_k = \varphi\left(W_g h_T^k + U_g\left(r_{(k)} \odot s_{k-1}\right)\right) \tag{19}$$

$$s_k = gz_{(k)} s_{k-1} + (1 - gz_{(k)})\tilde{s}_k \tag{20}$$

$$k = 1,2,\cdots,K \tag{21}$$

where $\boldsymbol{W_r}$, $\boldsymbol{U_r}$, $\boldsymbol{W_{gz}}$, $\boldsymbol{U_{gz}}$, $\boldsymbol{W_g}$, $\boldsymbol{U_g}$ are trainable parameters, $\boldsymbol{\sigma}$ and $\boldsymbol{\varphi}$ are activation functions, and ***K*** is a hyperparameter.

The final prediction of the output variables is given by:

$$\hat{y}_{k+K} = W_{LSTM} h_T^{k+K} + W_{GRU} s_{k+K} \tag{22}$$

where $\boldsymbol{W_{LSTM}}$ and $\boldsymbol{W_{GRU}}$ are trainable parameters. To prevent overfitting on the training set, regularization and dropout modules were incorporated into both the LSTM and GRU models.

2.3 Loss Function Design

For general dynamic modeling tasks, the loss function is typically defined as the difference between the predicted output variables and the actual value. In the proposed AR-SAE-RNN model, to effectively integrate input reconstruction and model regression, the loss function includes not only the prediction error of the output variables but also the reconstruction error of the input:

$$l = \alpha \frac{1}{N_{Train} T} \sum_{k=1}^{N_{Train}} \sum_{t=1}^{T} \left(\tilde{x}_t^k - x_t^k \right)^2 + \frac{1}{N_{Train}} \sum_{k=1}^{N_{Train}} (\hat{y}_k - y_k)^2 \quad (23)$$

where Γ is the hyperparameter representing the weight of the input reconstruction, and $\boldsymbol{N_{Train}}$ denotes the number of training samples. The calculation model for the reconstructed input $\tilde{\boldsymbol{x}}_{\boldsymbol{t}}^{\boldsymbol{k}}$ can be referenced from the literature **[6]**.

2.4 Model Training and Evaluation

The proposed AR-SAE-RNN model is relatively complex, and training all the parameters simultaneously may lead to divergence. Therefore, a distributed training strategy is adopted: the model parameters are trained sequentially in the order of SAE, LSTM, and GRU. At each step, only the parameters of one module are trained while freezing the others. After all modules are pre-trained, a fine-tuning process is applied to jointly optimize all parameters. Notably, distributed training is also required for the SAE module, and the specific method can be referred to in the literature **[6]**.

After training, the model performance is evaluated using two widely accepted metrics: root mean square error (RMSE) and the coefficient of determination (R^2):

$$RMSE = \sqrt{\frac{1}{N_{Train}} \sum_{k=1}^{N_{Train}} (\hat{y}_k - y_k)^2} \quad (24)$$

$$R^2 = 1 - \frac{\sum_{k=1}^{N_{Train}} (\hat{y}_k - y_k)^2}{\sum_{k=1}^{N_{Train}} (\overline{y_k} - y_k)^2} \quad (25)$$

where $\overline{\boldsymbol{y_k}}$ represents the mean value of the output variables in the training set.

3. INDUSTRIAL APPLICATION

The industrial case study presented in this paper focuses on the distillation column in the product separation section of the production process, as shown in Fig. 2. In this distillation column, the bottom product stream is composed of heavy key components with a high purity. The top product stream consists of a mixture of various components, which undergo secondary separation downstream.

The purity of the heavy key components in the top liquid product directly influences the purity of the bottom product and the difficulty of the downstream secondary separation. This is a critical indicator in the system, referred to as HKC. Therefore, this study constructs a dynamic model with other process variables as inputs and the HKC purity as the output, aiming to achieve real-time soft sensing of HKC purity.

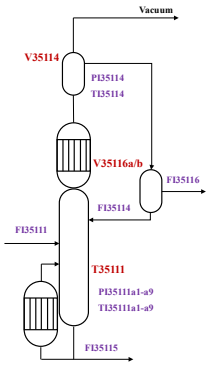

Figure 2. Separation Distillation Column.

As shown in Fig. 2, all process variables were sampled at a frequency of 1 minute. The HKC purity was measured online using Raman spectroscopy at a frequency of 10 minutes, which is costly. We collected 70 days of process data for model development. We first applied the 3σ rule to remove outlier samples. For the remaining valid samples, we used cubic spline interpolation to impute missing values in the process data. Finally, to ensure stable training of the neural network model, both the process data and output variables were normalized. Then, the dataset was divided into training, validation, and test sets in a ratio of 5:1:4.

To evaluate the performance of the AR-SAE-RNN model, the standard LSTM model was used as the baseline. Additionally, SAE-LSTM and LSTM-GRU models were included to assess the individual contributions of each module to prediction accuracy. All hyperparameters mentioned in this paper, including training-related parameters such as learning rate, were optimized through grid search.

The prediction performance on the test set is shown in Fig. 3, while the RMSE, R^2, and training time are summarized in Table 1. The results of hyperparameter tuning are summarized in Table 2. The standard LSTM model, benefiting from a large amount of training data, achieved accurate predictions for overall trends. However, its local prediction accuracy was suboptimal, with significant errors observed in many regions. By incorporating input dimensionality reduction through SAE, as well as temporal modeling of the HKC purity across samples using GRU, the AR-SAE-RNN model not only maintained accurate overall trend predictions but also significantly improved local prediction accuracy, achieving a 25% improvement in RMSE. Additionally, LSTM is designed to capture long-term dependencies but lacks inherent noise-filtering mechanisms, meaning it learns and propagates both useful patterns and noise through its hidden states. In contrast, the AR-SAE-RNN model integrates a SAE and a GRU, both of which contribute to noise reduction. The SAE compresses and extracts key features from the input data, filtering out irrelevant fluctuations before they

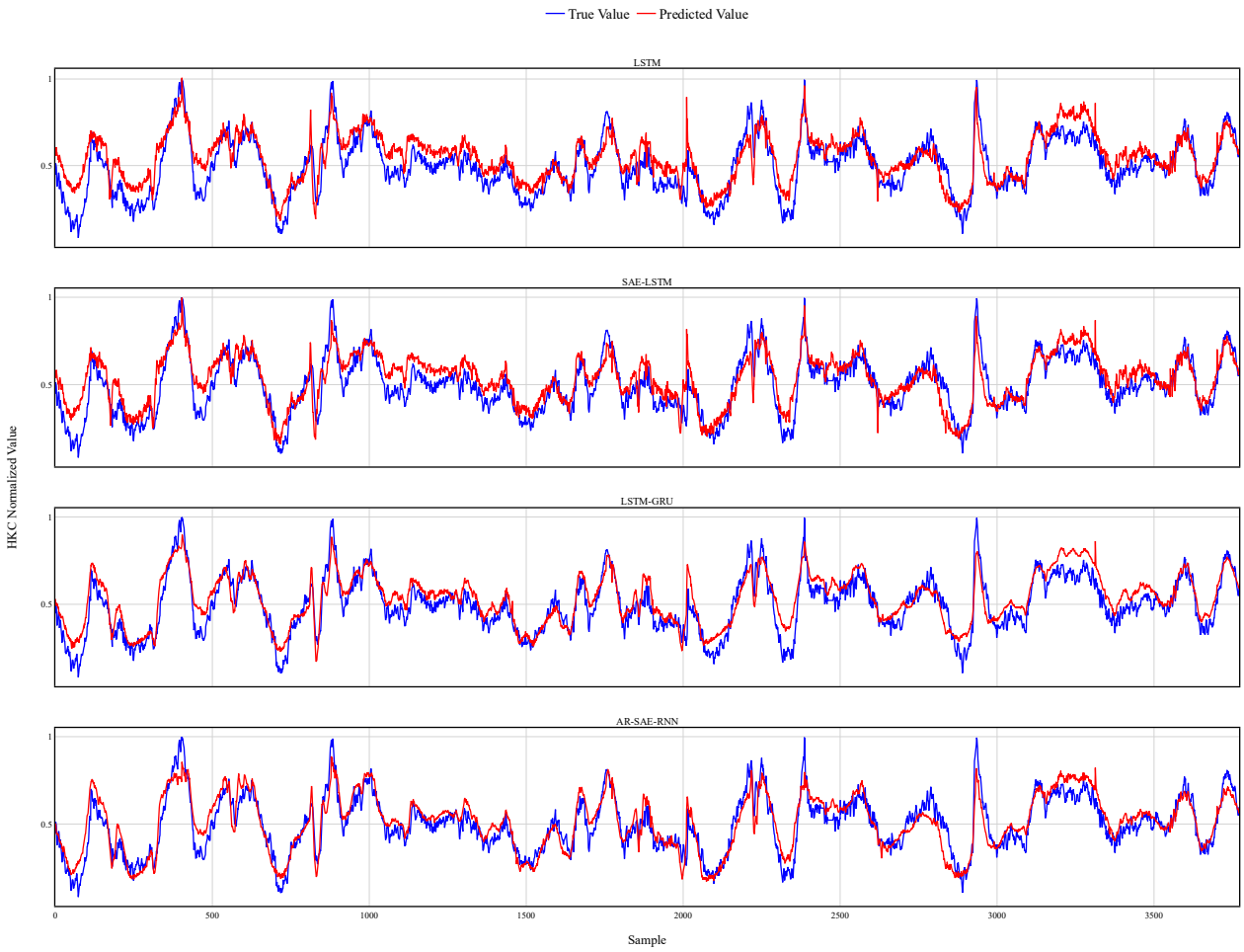

Figure 3. Comparison of predicted results and ground truth among four models.

reach the recurrent layers. Additionally, the GRU, with its simpler gating mechanism, is more resistant to noise accumulation, refining the hidden state representations and further attenuating residual noise. As a result, the AR-SAE-RNN model produces smoother and more robust predictions, making it more effective for handling noisy industrial data.

Table 1: Summary of Results for Four Models.

Model	RMSE	R^2	Training-Time(min)
LSTM	0.09636	0.7142	0.25
SAE-LSTM	0.09006	0.7503	0.97
LSTM-GRU	0.07764	0.8144	0.45
AR-SAE-RNN	**0.07262**	**0.8377**	**1.38**

Table 2: Hyperparameter Results.

Hyperparameter	Result
E	3
T	4
K	14
Γ	0.3
Lr(Pre-Trained)	1e-3
Lr(Fine-Tuning)	1e-6
Batch	400
LSTM Hidden Size	64
GRU Hidden Size	32
Epoch(SAE)	200
Epoch(LSTM)	200
Epoch(GRU)	70
Epoch(Fine-Tuning)	50
Dropout	0.2

From Table 1 and Fig. 3, it is evident that the GRU module has the most significant impact on improving prediction accuracy. This underscores the importance of considering the autocorrelation of output variables in dynamic modeling. Furthermore, the proposed model does not rely on the actual values of past output variables as inputs, making it more practical for industrial applications.

4. CONCLUSION

To address the challenges of dynamic modeling, the proposed AR-SAE-RNN model introduces a novel approach compared to traditional RNN models by effectively accounting for the autocorrelation of output variables. The model integrates temporal modeling of output variables across samples while leveraging SAE to reconstruct input features. This unified framework has demonstrated excellent performance in the soft sensing case study of the distillation column.

Moreover, we aim to enhance our model in three key aspects as part of our future work. First, we will upgrade the SAE to a Variational Autoencoder (VAE), which encodes data as a probability distribution rather than a deterministic value, improving robustness by filtering noise in abnormal process data. Second, we will incorporate an attention mechanism into the GRU modeling of output variable autocorrelation, allowing the model to focus on the most relevant hidden states and improving predictive accuracy. Finally, we aim to integrate a mechanistic model for the distillation unit by constructing a simplified dynamic wave model that captures the temporal and spatial distribution of component compositions along the column height. This framework will decompose the temporal network into three components: sequential modeling of process data, autocorrelation modeling of output variables, and spatial correlation modeling of output variables, which will be combined to enhance predictive performance.

ACKNOWLEDGEMENTS

The financial support of the "Pioneer" and "Leading Goose" R&D Program of Zhejiang (Grant No. 2024C01028) and the State Key Laboratory of Industrial Control Technology, China (Grant No. ICT2024C04) are gratefully acknowledged.

REFERENCES

1. Sun Q Q, Ge Z Q. A Survey on Deep Learning for Data-Driven Soft Sensors. *IEEE Transactions on Industrial Informatics*, 17(9):5853-5866 (2021). https://doi.org/10.1109/TII.2021.3053128.
2. Rumelhar D E, Hinton G E, Williams R J. Learning Representations by Back Propagating Errors. *Nature*, 323(6088):533-536 (1986). https://doi.org/10.1038/323533a0.
3. You Y, Nikolaou M. Dynamic process modeling with recurrent neural networks. *AIChE Journal*, 39(10):1654-1667 (1993). https://doi.org/10.1002/aic.690391009.
4. Hochreiter S, Schmidhuber J. Long Short-Term Memory. *Neural Computation*, 9(8):1735-1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735.
5. Cho K, Merrienboer B V, Gulcehre C, et al. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *Computer Science*, 1724–1734 (2014). https://doi.org/10.3115/v1/D14-1179.
6. Yuan X, Zhou J, Huang B, et al. Hierarchical Quality-Relevant Feature Representation for Soft Sensor Modeling: A Novel Deep Learning Strategy. *IEEE Transactions on Industrial Informatics*, 2020, 16(6):3721-3730 (2020). https://doi.org/10.1109/TII.2019.2938890.
7. Peng X Y, Wang Y L, Liu C L, et al. Dynamic-static collaborative strategy for industrial data modeling based on hierarchical deep networks. Measurement Science and Technology, 33(12):125010-1-125010-12 (2022). https://doi.org/10.1088/1361-6501/ac86e6.
8. Zhang H, Tang Z H, Xie Y F, et al. ES-net: An Integration Model Based on Encoder–Decoder and Siamese Time Series Difference Network for Grade Monitoring of Zinc Tailings and Concentrate. IEEE Transactions on Industrial Electronics, 70(11):11819-11830 (2023). https://doi.org/10.1109/TIE.2022.3227274.
9. Ma Y, Li H G. GRU-Auto-Encoder neural network based methods for diagnosing abnormal operating conditions of steam drums in coal gasification plants. Computers & Chemical Engineering, 143:107097 (2020). https://doi.org/10.1016/j.compchemeng.2020.107097.
10. Smagulova K, James A P. A survey on LSTM memristive neural network architectures and applications. The European Physical Journal Special Topics, 228:2313-2324 (2019). https://doi.org/10.1140/epjst/e2019-900046-x.
11. Xie C R, Yao R J, Zhu L Y. Soft-Sensor Development through Deep Learning with Spatial and Temporal Feature Extraction of Complex Processes. Industrial & Engineering Chemistry Research, 62:519-534 (2023). https://doi.org/10.1021/acs.iecr.2c03137.
12. Yuan X F, Li L, Wang Y L. Nonlinear Dynamic Soft Sensor Modeling With Supervised Long Short-Term Memory Network. IEEE Transactions on Industrial Informatics, 16(5):3168-3176. https://doi.org/10.1109/TII.2019.2902129.