Deep Cascade Gradient RBF Networks With Output-Relevant Feature Extraction and Adaptation for Nonlinear and Nonstationary Processes

Tong Liu, *Member, IEEE*, Zeyue Tian, Sheng Chen, *Fellow, IEEE*, Kai Wang, and Chris J. Harris

***Abstract*—The main challenge for industrial predictive models is how to effectively deal with big data from high-dimensional processes with nonstationary characteristics. Although deep networks, such as the stacked autoencoder (SAE), can learn useful features from massive data with multilevel architecture, it is difficult to adapt them online to track fast time-varying process dynamics. To integrate feature learning and online adaptation, this article proposes a deep cascade gradient radial basis function (GRBF) network for online modeling and prediction of nonlinear and nonstationary processes. The proposed deep learning method consists of three modules. First, a preliminary prediction result is generated by a GRBF weak predictor, which is further combined with raw input data for feature extraction. By incorporating the prior weak prediction information, deep output-relevant features are extracted using a SAE. Online prediction is finally produced upon the extracted features with a GRBF predictor, whose weights and structure are updated online to capture fast time-varying process characteristics. Three real-world industrial case studies demonstrate that the proposed deep cascade GRBF network outperforms existing state-of-the-art online modeling approaches as well as deep networks, in terms of both online prediction accuracy and computational complexity.**

***Index Terms*—Deep learning, gradient radial basis function (GRBF) network, high-dimensional and nonstationary processes, online adaptation, output-relevant features, stacked autoencoder (SAE).**

I. Introduction

TO ACHIEVE energy efficiency and operational effectiveness, as well as to maintain safety for industrial processes, it is essential to carry out real-time process control, optimization, and monitoring [1]. This critically depends on timely identification and prediction of key process variables. However, due to process drifts resulting from operating condition changes, equipment aging, or catalyst deactivation, etc., most industrial processes exhibit severe nonstationary characteristics [2]–[5]. Time-varying process dynamics impose the need that nonlinear predictive models must have online adaptation capacity [6]–[8]. Another serious problem encountered in industry is that abundant process data are often high dimensional with strong correlations and redundancies. This may lead to instability and poor robustness, as well as unsatisfactory performance of predictive models. Hence, effective features that contain essential and compressed data information should be extracted first during process modeling [9]–[12]. This is often achieved by a deep network with multilevel feature extraction layers. However, integrating feature extraction and online adaptation into one model is very challenging, since it is prohibitive to adapt large deep networks within a small sampling period in order to track fast time-varying process dynamics. This motivates our current work to develop an effective model that combines both feature extraction and online adaptation.

The radial basis function (RBF) network as a shallow learning model has found wide-ranging applications in diverse engineering fields [13]–[17]. With a set of nonlinear kernels imposed on training input data, the orthogonal least squares (OLS) learning can be applied to construct a compact RBF model [18]–[20]. This procedure can be interpreted as encoding the process's nonlinear dynamics in the hidden layer nodes, with each RBF node storing an independent process state. To provide some adaptive capability, the RBF network can update its weight vector using some adaptive estimators, such as the recursive least square (RLS) [21]–[23]. However, during the online operation of nonstationary processes, the process dynamics can vary dramatically and new process states may appear. In order to capture the newly emerged process state, the model structure should also be updated in real time. An effective approach to achieve this goal is the fast tunable RBF (TRBF) [24]. Starting with an initial compact RBF model, the TRBF method adjusts the RBF nodes as well as the weights online to adaptively modeling nonstationary data. The experimental results of [24] show that this TRBF outperforms many state-of-the-art online modeling approaches for nonstationary data.

For nonstationary time series involving variations of local mean and trend, the series can be made stationary by applying

Manuscript received 14 April 2021; revised 26 August 2021 and 3 January 2022; accepted 12 February 2022. Date of publication 3 March 2022; date of current version 20 July 2023. This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFB1312000, and in part by the Fundamental Research Funds for the Central Universities under Grant 2019CDYGZD010. This article was recommended by Associate Editor P. Shi. *(Corresponding authors: Tong Liu; Kai Wang.)*

Tong Liu is with the Department of Computer Science, University of Sheffield, Sheffield S1 4DP, U.K. (e-mail: t.liu@sheffield.ac.uk).

Zeyue Tian is with the School of Engineering, Hong Kong University of Science and Technology, Hong Kong, China (e-mail: ztianad@connect.ust.hk).

Sheng Chen and Chris J. Harris are with the School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, U.K. (e-mail: sqc@ecs.soton.ac.uk; chrisharris57@msn.com).

Kai Wang is with the School of Automation, Chongqing University, Chongqing 400044, China (e-mail: kaiwang@cqu.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCYB.2022.3152107.

Digital Object Identifier 10.1109/TCYB.2022.3152107

a difference operation on the raw data [25]. By integrating a similar mechanism of differencing the original time series data into the RBF network and modifying each hidden node as a local predictor, the gradient RBF (GRBF) network was proposed for nonlinear and nonstationary time-series prediction [26]. The OLS learning can also be applied to construct a compact GRBF model from the training data. For time series with highly time-varying characteristics other than variations of local mean and trend, a fast adaptive GRBF (AGRBF) algorithm was proposed for online time-series modeling and prediction [27]. A recent work [28] has extended this AGRBF to online modeling and prediction of nonlinear and nonstationary dynamic processes. Similar to the TRBF [24], during online operation when the error for the current data is unacceptable, the AGRBF adapts the model structure by replacing the worst node with a new node to encode the new data. Due to the local prediction property of GRBF node, this new node optimization is much more efficient than the TRBF, and it imposes little online computation. This AGRBF outperforms the TRBF, in terms of both online modeling accuracy and computational complexity [28].

Beyond the aforementioned shallow networks, deep learning has gained growing influence in machine learning [29], [30]. Due to the deep architecture with multilevel nonlinearities, deep learning can learn hierarchical feature representations effectively for large-scale complex data. Since deep networks are good at discovering intricate data patterns through feature learning, they have been used for big data modeling and achieved excellent results in the industrial field [31]–[33]. Deep learning methods for process modeling typically include two models: 1) the recurrent neural network (RNN) or its long short-term memory (LSTM) variant and 2) the stacked autoencoder (SAE). RNN-like models are designed to extract dynamic temporal information from data, and they have been successfully applied to many prediction problems [34], [35]. However, with multiple loops and gates, the complex structure of RNN deters its use in online modeling task. The SAE is generally used to extract latent useful features from massive data, and a regression model is then developed based on the extracted features for prediction [36]. Since SAE can learn more complex and abstract features with a hierarchical structure, it can provide a better approximation for complex nonlinear systems. Several works have improved the performance of the SAE by incorporating quality-relevant information [9], [31], [37] or data augmentation [38] to enhance feature representation. To our best knowledge, the most improvements in SAE are from a feature learning perspective, and applying SAE in nonstationary environments for online modeling remains largely understudied. This is particularly challenging, since adapting a deep model online is computationally very expensive, and it is impossible to optimize the SAE's structure in real time in order to track fast time-varying process characteristics.

Since optimizing a deep model structure online is impossible, an alternative solution is sought in this article, in order to develop an adaptive deep model for online modeling and prediction of nonlinear and nonstationary processes. Specifically, we propose a deep cascade GRBF network that integrates seamlessly the fast GRBF model adaptation and deep layerwise feature extraction. The proposed method consists of three components, namely: 1) the GRBF weak predictor; 2) the SAE feature extraction; and 3) the GRBF adaptive predictor. These three parts are connected in series. First, a GRBF model is trained to produce a preliminary prediction of the target value, which is called the weak prediction. This GRBF weak predictor is a shallow network and its task is to provide the prior output information to the SAE. After combining the weak prediction with the raw input data, the new expanded input vector that contains the output information is ready for feature extraction. By stacking multiple autoencoders (AEs), high-level output-relevant features are progressively learned from their previous low-level ones layer by layer. Finally, the extracted deep output-relevant features are fed into an AGRBF predictor, whose weights as well as model structure are updated online to capture time-varying process characteristics. Our novel contributions can be summarized as follows.

1) To effectively deal with high-dimensional and nonstationary data, a deep cascade GRBF network is proposed, which integrates output-relevant feature learning and online adaptation naturally.
2) The proposed scheme with very deep architecture is computationally very efficient for online model adaptation to track time-varying process characteristics, and it does not require any complicated online structure optimization.
3) Extensive results demonstrate that our deep cascade GRBF network outperforms existing state-of-the-art online adaptive models as well as deep-learning models for nonlinear and nonstationary data prediction.

The remainder of this article is organized as follows. Section II reviews the related algorithms, including the GRBF network and its online adaptive mechanism, as well as the SAE feature extraction. Section III presents the proposed deep cascade GRBF network in detail. Section IV evaluates the proposed method with three case studies. Finally, Section V concludes this article with remarks about future works.

II. Revisit of Related Algorithms

A. GRBF Neural Network

The GRBF neural network is an effective tool for modeling nonlinear and nonstationary data [26]–[28]. Consider a nonlinear and nonstationary process with n_i system inputs $\boldsymbol{u}_t = [u_{1,t} \cdots u_{n_i,t}]^T \in \mathbb{R}^{n_i}$ having input lag n_u and the system output $y_t \in \mathbb{R}$ having output lag n_y. Assume that we have collected a training data set composed by $\{\boldsymbol{x}_t, d_t; y_t\}_{t=1}^{N}$, where N is the number of training samples, and

$$\boldsymbol{x}_t = \left[(y_{t-1} - y_{t-2}) \cdots \left(y_{t-n_y+1} - y_{t-n_y}\right) \boldsymbol{u}_{t-1}^T \cdots \boldsymbol{u}_{t-n_u}^T\right]^T \quad (1)$$

denotes the input vector to the GRBF model, while

$$d_t = y_t - y_{t-1} \quad (2)$$

is the system output gradient. Observe that the differenced past outputs rather than the past outputs form the part of the input vector $\boldsymbol{x}_t$. If the system output lag is $n_y = 0$ or no output information is provided, the input vector is reduced to $\boldsymbol{x}_t = [\boldsymbol{u}_{t-1}^T \cdots \boldsymbol{u}_{t-n_u}^T]^T$, while for time series $n_u = 0$ and the input vector becomes $\boldsymbol{x}_t = [(y_{t-1} - y_{t-2}) \cdots (y_{t-n_y+1} - y_{t-n_y})]^T$.

Like the classic RBF node, the Gaussian function typically serves as the GRBF node's nonlinearity. The main difference is that the response of a GRBF node is further multiplied by an additional term $(y_{t-1} + \delta)$. Hence, the response of the jth GRBF hidden node to the input vector $\boldsymbol{x}_t$ is given by

$$p_j(\boldsymbol{x}_t) = \exp\left(\frac{-\|\boldsymbol{x}_t - \boldsymbol{c}_j\|^2}{2\sigma^2}\right) \times \left(y_{t-1} + \delta_j\right) \tag{3}$$

where σ is the width of Gaussian kernel, $\boldsymbol{c}_j$ is the node center, and δ_j is a scalar associated with the hidden node. The width σ can be set to the maximum Euclidean distance among nodes, and the term $(y_{t-1}+\delta_j)$ can be interpreted as a local prediction of y_t by the jth hidden node [28]. From (3), if the input vector is very similar to the jth center, the value of the jth Gaussian function is close to 1 and the predictor $(y_{t-1} + \delta_j)$ becomes fully active. Let M be the number of hidden nodes. The GRBF network can then be formulated as the linear combination of its hidden layer's response to model y_t as

$$y_t = \sum_{j=1}^{M} p_j(\boldsymbol{x}_t)\theta_j + \xi_t \tag{4}$$

where $\boldsymbol{\theta} = [\theta_1 \cdots \theta_M]^T$ is the weight vector, and ξ_t is a zero-mean model residual sequence.

A compact M-term GRBF network with $M \ll N$ can readily be constructed from the training data set $\{\boldsymbol{x}_t, d_t; y_t\}_{t=1}^N$ using the OLS algorithm [18], [19]. In particular, if $\boldsymbol{x}_t$ is selected as the jth center $\boldsymbol{c}_j$, we set $\delta_j = d_t$ to ensure that the jth hidden node is a perfect local predictor of y_t [27], [28]. In this way, the problem of constructing a GRBF network is equivalent to the task of selecting an M-term subset model $\{\boldsymbol{c}_j, \delta_j\}_{j=1}^M$ from the full N-term model $\{\boldsymbol{x}_t, d_t\}_{t=1}^N$. The forward OLS selection algorithm selects a subset of M centers $\boldsymbol{c}_j$ and scalars δ_j one by one from the full model. At each step, a candidate with the maximum error reduction ratio (ERR) is chosen. The selection procedure is terminated when some termination criterion is met, yielding to an M-term subset model $\{\boldsymbol{c}_j, \delta_j\}_{j=1}^M$. Then, the weight vector of the selected M-term subset GRBF network can readily be solved by the backward substitution. The details of the OLS model selection based on the ERR criterion can be found in [18]–[20].

B. Online Adaptive Learning of GRBF Neural Network

During online operation, the weight vector of the GRBF model can be updated using the RLS algorithm to provide some tracking capability for time-varying processes. However, for highly nonstationary processes, this is insufficient, and the structure or the hidden layer of the GRBF model needs to be adapted to encode the newly emerging process state [28]. Thus, during online operation, the residual error of the GRBF network is monitored. Specifically, let $\boldsymbol{p}_t = [p_1(\boldsymbol{x}_t) \cdots p_M(\boldsymbol{x}_t)]^T$ denote the hidden layer response vector for the given input $\boldsymbol{x}_t$ and $\boldsymbol{\theta}_{t-1} = [\theta_{1,t-1} \cdots \theta_{M,t-1}]^T$ be the weight vector obtained at the previous sample. Then, the residual for the prediction of y_t based on the current model is given by

$$e_t = y_t - \boldsymbol{p}_t^T \boldsymbol{\theta}_{t-1}. \tag{5}$$

The model performance can be measured by the cost

$$\widetilde{e}_t = \frac{e_t^2}{y_t^2} \tag{6}$$

with respect to a given threshold ε.

If $\widetilde{e}_t < \varepsilon$, then the model structure is unchanged and only the weight vector is updated using the RLS algorithm

$$\begin{cases} \boldsymbol{k}_t = \boldsymbol{\Gamma}_{t-1}\boldsymbol{p}_t\left(\gamma + \boldsymbol{p}_t^T \boldsymbol{\Gamma}_{t-1}\boldsymbol{p}_t\right)^{-1} \\ \boldsymbol{\Gamma}_t = \left(\boldsymbol{\Gamma}_{t-1} - \boldsymbol{k}_t \boldsymbol{p}_t^T \boldsymbol{\Gamma}_{t-1}\right)\gamma^{-1} \\ \boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + \boldsymbol{k}_t e_t \end{cases} \tag{7}$$

where $\boldsymbol{k}_t \in \mathbb{R}^M$ is the Kalman gain vector, $0.9 \le \gamma < 1$ is the forgetting factor, and the inverse of covariance matrix $\boldsymbol{\Gamma}_t \in \mathbb{R}^{M\times M}$ is initialized to $\boldsymbol{\Gamma}_0 = \vartheta \boldsymbol{I}_M$ in which ϑ is a large positive constant and $\boldsymbol{I}_M$ is the $M \times M$ identity matrix.

If $\widetilde{e}_t \ge \varepsilon$, the model performs inadequately. Thus, the network structure needs to be updated according to the newly emerged process state by replacing the worst performing node with a new node [28]. The node contribution to the model performance is measured by the weighted node-output variance (WNV) defined by

$$\text{WNV}_j = \left(\theta_{j,t-1} p_j(\boldsymbol{x}_t)\right)^2, \ \ 1 \le j \le M. \tag{8}$$

We can compare the nodes' WNVs and select the one with the smallest value as the worst performing node. Let

$$m = \arg \min_{1 \le j \le M} \text{WNV}_j. \tag{9}$$

Then, the mth node is the worst node and is replaced by a new one. Since the goal of the new replacement node is to encode the newly emerged process state, we can simply set the new node center as $\boldsymbol{c}_m = \boldsymbol{x}_t$ and scalar as $\delta_m = y_t - y_{t-1}$ to ensure that the new mth node is a perfect local predictor of y_t. Because the set of centers now contains a new one, the width of the Gaussian response σ is recalculated based on the maximum Euclidean distance among the centers.

After the new hidden node is determined, the weight vector of the new GRBF network is calculated by the regularized least square (LS) estimator as

$$\boldsymbol{\theta}_t = \left(\widetilde{\boldsymbol{p}}_t \widetilde{\boldsymbol{p}}_t^T + \lambda \boldsymbol{I}_M\right)^{-1} \widetilde{\boldsymbol{p}}_t y_t \tag{10}$$

where $\widetilde{\boldsymbol{p}}_t$ is the new hidden layer response vector after the new node replacement, and λ is a positive small regularization parameter. After the regularized LS estimation (10), the inverse covariance matrix is reinitialized according to

$$\boldsymbol{\Gamma}_t = \left(\widetilde{\boldsymbol{p}}_t \widetilde{\boldsymbol{p}}_t^T + \lambda \boldsymbol{I}_M\right)^{-1} \tag{11}$$

to ensure a smooth transition from one mode to another at the next sample. The threshold ε is the only algorithmic parameter of this AGRBF algorithm.

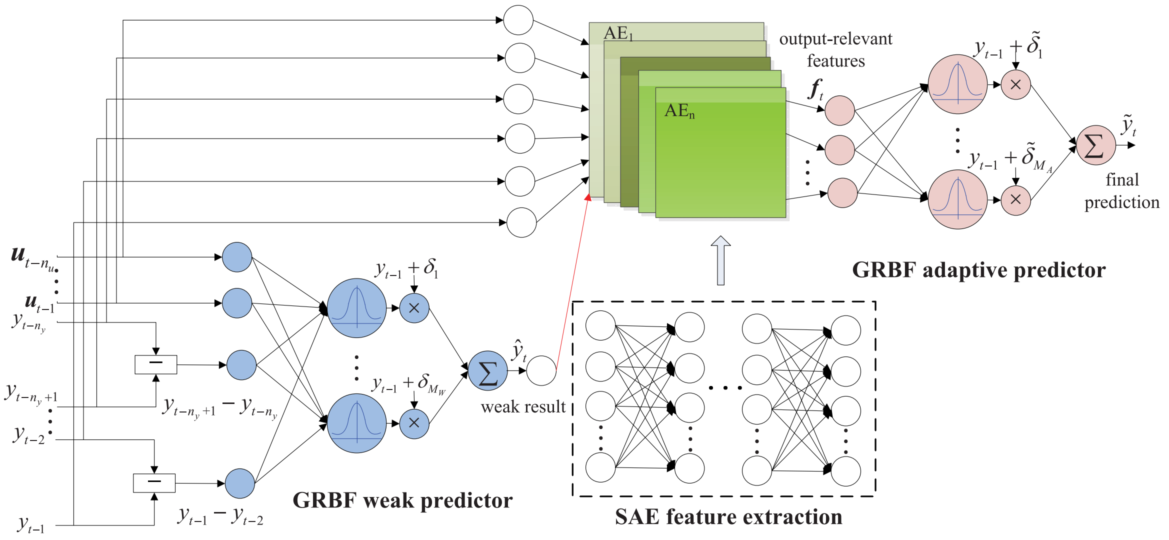


Fig. 1. Schematic of deep cascade GRBF network.

C. SAE

SAE is a deep network with hierarchical multiple AEs. Each AE is a three-layer unsupervised self-learning network with encoder and decoder. Let the inputs of the AE be $\boldsymbol{x}'_t$. The encoder projects $\boldsymbol{x}'_t$ from the input layer onto the hidden layer $\boldsymbol{h}_t = [h_1(\boldsymbol{x}'_t) \cdots h_s(\boldsymbol{x}'_t)]^T$ by the nonlinear mapping $\boldsymbol{f}$ as

$$\boldsymbol{h}_t = \boldsymbol{f}\left(\boldsymbol{W}\boldsymbol{x}'_t + \boldsymbol{b}\right) \tag{12}$$

where s is the dimension of the hidden layer, and $\boldsymbol{W}$ and $\boldsymbol{b}$ are the weight matrix and bias vector, respectively, connecting the input layer to the hidden layer. The decoder reconstructs the input vector $\boldsymbol{x}'_t$ by mapping $\boldsymbol{h}_t$ onto the output layer as

$$\tilde{\boldsymbol{x}}'_t = \tilde{\boldsymbol{f}}\left(\tilde{\boldsymbol{W}}\boldsymbol{h}_t + \tilde{\boldsymbol{b}}\right) \tag{13}$$

where $\tilde{\boldsymbol{f}}$ is the output layer's nonlinear mapping, and $\tilde{\boldsymbol{W}}$ and $\tilde{\boldsymbol{b}}$ are the connecting weight matrix and bias vector, respectively, from the hidden layer to the output layer. From (12) and (13), the task of AE is to learn a mapping $\boldsymbol{F}(\boldsymbol{x}'_t) = \tilde{\boldsymbol{f}}(\boldsymbol{f}(\boldsymbol{x}'_t)) \approx \boldsymbol{x}'_t$ that keeps the reconstructed output $\tilde{\boldsymbol{x}}'_t$ as similar as possible to the original input $\boldsymbol{x}'_t$. Denote the training input data as $\boldsymbol{x}'_t \in \{\boldsymbol{x}'_1, \ldots, \boldsymbol{x}'_N\}$, where N is the number of training samples, and the corresponding features and reconstructed input data as $\boldsymbol{h}_t \in \{\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N\}$ and $\tilde{\boldsymbol{x}}'_t \in \{\tilde{\boldsymbol{x}}'_1, \ldots, \tilde{\boldsymbol{x}}'_N\}$, respectively. To obtain the model parameters $\{\boldsymbol{W}, \tilde{\boldsymbol{W}}, \boldsymbol{b}, \tilde{\boldsymbol{b}}\}$, the AE is trained by minimizing the mean squared reconstructed error

$$J_{\text{unsup}}\left(\boldsymbol{W}, \tilde{\boldsymbol{W}}, \boldsymbol{b}, \tilde{\boldsymbol{b}}\right) = \frac{1}{2N}\sum_{t=1}^{N}\left\|\tilde{\boldsymbol{x}}'_t - \boldsymbol{x}'_t\right\|^2 \tag{14}$$

using the gradient descend algorithm.

Multiple n AEs can be hierarchically stacked to construct a deep SAE network. Training the SAE involves the layer-wise unsupervised pretraining and supervised fine-tuning. In pretraining, the first AE maps the raw input data onto its hidden-layer features by minimizing the reconstruction error. After the first AE is trained, its hidden layer parameters $\{\boldsymbol{W}_1, \boldsymbol{b}_1\}$ are fixed, and the obtained hidden layer features $\boldsymbol{h}_{\text{AE},1}$ serve as the input to the second AE. Then, the second AE is trained to obtain its hidden layer parameters $\{\boldsymbol{W}_2, \boldsymbol{b}_2\}$ and the associated features $\boldsymbol{h}_{\text{AE},2}$. In a progressive way, the entire SAE is pretrained layer by layer until the last (nth) AE is obtained.

After the unsupervised pretraining, a regression layer with single output neuron having the weight vector $\boldsymbol{w}_\text{o}$ and bias b_o is added on the top of the SAE to produce the prediction $\tilde{y}_t$ of the process output y_t based on supervised learning. The entire network is fine-tuned by the backpropagation with the training data $\{\boldsymbol{x}'_t; y_t\}_{t=1}^N$ based on the cost function

$$J_{\text{sup}}(\boldsymbol{w}_\text{o}, b_\text{o}, \boldsymbol{W}_i, \boldsymbol{b}_i, 1 \le i \le n) = \frac{1}{2N}\sum_{t=1}^{N}(\tilde{y}_t - y_t)^2 \tag{15}$$

with the pretrained SAE's parameters used to initialize the hidden layers $\{\boldsymbol{W}_i, \boldsymbol{b}_i\}_{i=1}^n$ of the supervised SAE.

Since multiple AEs already provide highly nonlinear features, adopting a linear regression layer in the supervised fine-tuning is sufficient, and there is no need to employ a nonlinear regression layer. The nonlinear output neuron in the regression layer would possibly slow down training and lead to premature convergence, both due to gradient modulation effects in backpropagation.

III. Deep Cascade Adaptive GRBF Network

As depicted in Fig. 1, the proposed deep cascade GRBF network consists of three parts: 1) the GRBF weak predictor; 2) the SAE for feature extraction; and 3) the GRBF adaptive predictor. In the first part, the raw input data are fed into a GRBF network trained as an initial or weak predictor of the process output. In the second part, by combining the weak prediction of the weak GRBF predictor and the raw input data into a new input vector, the SAE is employed to extract its useful features. Through layerwise feature extraction, deep output-relevant features are obtained hierarchically. Finally, the output-relevant features are fed into an AGRBF predictor to make the accurate prediction adaptively. The advantage of this deep cascade GRBF network is twofold.

First, the weak predictor provides a preliminary prediction of the target value. This output information is incorporated into the layerwise feature extraction to extract the deep output-relevant features by the SAE. It is well known that enhanced nonlinear feature extraction can be achieved by incorporating quality-relevant information [9], [31], [37] or data augmentation [38] to the SAE. Explicitly, by incorporating the output information, namely, y_t, into the input to the SAE, it can extract better-quality nonlinear features. However, in our predictive modeling application, the current output y_t is unknown. In fact, given the input $\boldsymbol{x}_t$, the sole purpose of the SAE is to extract the nonlinear features for predicting or modeling the unknown current output y_t. In order to provide the SAE with the output-relevant information, we use the "second-best," that is, we provide the SAE with an estimate $\widehat{y}_t$ for the unknown y_t. This is achieved by the first GRBF predictor. It is called the "weak" predictor for the reason that its prediction $\widehat{y}_t$ is not our final prediction for the current output y_t. But this weak predictor provides the essential and vital output-relevant information to the SAE feature extractor.

Second, the proposed method can handle severe nonstationarity in the process data well. This capability comes from the GRBF adaptive predictor, not from the GRBF weak predictor and the SAE feature extractor. True, the GRBF weak predictor differences the output variable in the raw input data and it does make the underlying process less nonstationary, since the difference operation removes the local mean and trend [25]. But this GRBF weak predictor is fixed after training and only acts as a part of the input to the SAE during online operation. The SAE is also fixed after training, as it is impossible to optimize the multiple deep AE layers of the SAE online. However, the GRBF adaptive predictor can track the fast time-varying underlying process characteristics by updating its weights and structure effectively. It can be seen that our deep cascade GRBF network consists of two learning phases: initial training as well as online prediction and adaptive modeling.

A. Training Deep Cascade GRBF Network

The construction of the deep cascade GRBF network involves three stages, namely: 1) training the GRBF weak predictor; 2) training the SAE; and 3) training the GRBF adaptive predictor, respectively. Both GRBF networks are constructed using the OLS algorithm, while the SAE is trained by an unsupervised pretraining and a supervised fine-tuning as discussed in Section II-C.

In the first stage, we have the training data set $\boldsymbol{D}_W = \{\boldsymbol{X}_{\text{tr}}, \boldsymbol{d}_{\text{tr}}; \boldsymbol{y}_{\text{tr}}\} = \{\boldsymbol{x}_t, d_t; y_t\}_{t=1}^{N_{\text{tr}}}$, where $\boldsymbol{x}_t$ and d_t are given by (1) and (2), respectively, and N_{tr} is the number of training data, while $\boldsymbol{X}_{\text{tr}} = [\boldsymbol{x}_1\ \boldsymbol{x}_2 \cdots \boldsymbol{x}_{N_{\text{tr}}}] \in \mathbb{R}^{n_{\text{tr}} \times N_{\text{tr}}}$, $\boldsymbol{y}_{\text{tr}} = [y_1\ y_2 \cdots y_{N_{\text{tr}}}]^T \in \mathbb{R}^{N_{\text{tr}}}$, and $\boldsymbol{d}_{\text{tr}} = [d_1\ d_2 \cdots d_{N_{\text{tr}}}]^T \in \mathbb{R}^{N_{\text{tr}}}$ are the input, desired output, and desired output difference data, respectively, with $n_{\text{tr}} = n_y - 1 + n_u n_i$ being the input dimension. A compact M_W-term GRBF network is constructed from the training set $\boldsymbol{D}_W$ using the OLS algorithm. With this trained GRBF weak predictor, the weak predictions $\widehat{\boldsymbol{y}}_{\text{tr}} = [\widehat{y}_1\ \widehat{y}_2 \cdots \widehat{y}_N]^T \in \mathbb{R}^{N_{\text{tr}}}$ are generated for the desired values $\boldsymbol{y}_{\text{tr}}$.

In the second stage, the training data set is expanded to $\boldsymbol{D}_F = \{\boldsymbol{X}'_{\text{tr}}; \boldsymbol{y}_{\text{tr}}\} = \{\boldsymbol{x}'_t; y_t\}_{t=1}^{N_{\text{tr}}}$ by including the weak prediction $\widehat{\boldsymbol{y}}_{\text{tr}}$ as the part of the input to the SAE, where the training input data $\boldsymbol{x}'_t$ for the SAE is defined as

$$\boldsymbol{x}'_t = \left[\widehat{y}_t\ y_{t-1} \cdots y_{t-n_y}\ \boldsymbol{u}_{t-1}^T \cdots \boldsymbol{u}_{t-n_u}^T\right]^T \in \mathbb{R}^{(n_{\text{tr}}+2)}. \tag{16}$$

The new training data $\boldsymbol{D}_F$ are fed into the SAE so as to learn the corresponding useful features. Specifically, the layerwise unsupervised pretraining is carried out from the first AE to the last AE by optimizing the objective function (14). After the pretraining, a linear regression output layer is added on the top of the last AE to form the supervised SAE. With the true target values $\boldsymbol{y}_{\text{tr}}$, the gradient descend optimization is employed to minimize the objective function (15), so as to fine-tune the entire SAE network. After the SAE is trained, the extracted features $\boldsymbol{F}_{\text{tr}}$ can be obtained from the last AE for the input $\boldsymbol{X}'_{\text{tr}}$. Then, the regression output layer is removed, and the last AE is connected to the GRBF adaptive predictor.

In the third stage, the extracted features $\boldsymbol{F}_{\text{tr}}$ by the SAE serve as the input to the GRBF adaptive predictor. Note that there is no differencing operation in the input layer of this GRBF network. The training data set $\boldsymbol{D}_A = \{\boldsymbol{F}_{\text{tr}}, \boldsymbol{d}_{\text{tr}}; \boldsymbol{y}_{\text{tr}}\}$ is utilized to construct a compact M_A-term GRBF model using the OLS algorithm. The output of this model provides the prediction of the process output. After this stage, the training of the deep cascade GRBF network is completed, and the trained deep cascade GRBF network is ready for online prediction and modeling.

B. Online Prediction and Adaptive Modeling

During online operation, it is impossible to optimize the entire deep cascade GRBF network within a small sampling period for tacking the fast time-varying process characteristics. Since the main focus of the first two components is feature learning to provide the essential and compressed input data for the third module, during online operation, it is sufficient to fix the structures and parameters of the weak GRBF predictor and the SAE, and only to adapt the final GRBF adaptive predictor for tracking the time-varying underlying dynamics between the extracted features and the process output.

Predicting Process Output: At sample time t, given the process's raw observations or the process input vector at t

$$\boldsymbol{x}_{p_t} = \left[y_{t-1} \cdots y_{t-n_y}\ \boldsymbol{u}_{t-1}^T \cdots \boldsymbol{u}_{t-n_u}^T\right]^T \in \mathbb{R}^{n_p} \tag{17}$$

the new input $\boldsymbol{x}_t$ for the GRBF weak predictor is formed, and the hidden layer response of the weak GRBF predictor $\boldsymbol{p}_{W_t}$ is obtained by the nonlinear mapping (3). The weak prediction result is then produced as $\widehat{y}_t = \boldsymbol{p}_{W_t}^T \boldsymbol{\theta}_W$, where $\boldsymbol{\theta}_W$ is the weight vector of the weak predictor. The weak prediction $\widehat{y}_t$ is further combined with the raw observations $\boldsymbol{x}_{p_t}$ to form the new input vector $\boldsymbol{x}'_t$ to the SAE. Through forward propagation from the first feature layer to the last one, the deep output-relevant features are extracted at the last AE as $\boldsymbol{f}_t$. The extracted features $\boldsymbol{f}_t$ serve as the input to the AGRBF predictor, whose hidden layer response vector $\boldsymbol{p}_{A_t}$ is obtained according to (3). Then, the model output, which is produced as $\widetilde{y}_t = \boldsymbol{p}_{A_t}^T \boldsymbol{\theta}_{A_{t-1}}$ with $\boldsymbol{\theta}_{A_{t-1}}$ being the weight vector obtained at sampling time $t-1$, provides the deep cascade GRBF network's prediction of the process output y_t.

Adapting Deep Cascade GRBF Network: When the observation of the process output y_t arrives, the newest output difference d_t also becomes available, and y_t and d_t are used with the input vector $\boldsymbol{f}_t$ to update the AGRBF predictor's structure and weights. Specifically, given the prediction $\widetilde{y}_t$ by the current GRBF predictor, the modeling performance, defined as $\widetilde{e}_t = (y_t - \widetilde{y}_t)^2/y_t^2$ of (5) and (6), is measured. If $\widetilde{e}_t < \varepsilon$, the model structure remains unchanged and only the current weight vector $\boldsymbol{\theta}_{A_{t-1}}$ is updated into the new one $\boldsymbol{\theta}_{A_t}$ with the RLS (7). If $\widetilde{e}_t \geq \varepsilon$, the node replacement takes place. The values of WNV_j, $1 \leq j \leq M_A$, are calculated using (8), and the worst performing node is identified by (9) as the mth node, which is then replaced by a new node. To be specific, the new mth node's center and scalar are set to $\boldsymbol{c}_m = \boldsymbol{x}_t$ and $\delta_m = y_t - y_{t-1}$, respectively, to ensure that it becomes a perfect local predictor of y_t. After the node replacement, the new weight vector of the GRBF predictor $\boldsymbol{\theta}_{A_t}$ is calculated by the regularized LS estimation (10), and the inverse covariance matrix $\boldsymbol{\Gamma}_t$ is updated with (11).

C. Algorithm Summary

The proposed deep cascade GRBF network is summarized in Algorithm 1. Our proposed algorithm is computationally very efficient during online operation. This is because the first two components of the cascade network are both fixed, and online complexity is determined by the AGRBF predictor. If the GRBF adaptive predictor performs only weight adaptation, the complexity comes from the RLS algorithm (7), which is on the order of $\mathcal{O}({M_A}^2)$, while if the node replacement occurs, the WNV calculation in (8) costs $\mathcal{O}(M_A)$ and the regularized LS estimator (10) has the complexity no more than $\mathcal{O}({M_A}^3)$. Thus, the online computational complexity per sample of the proposed algorithm is no more than $\mathcal{O}({M_A}^3)$, which is clearly affordable as M_A is very small.

We expect that the online complexity of the proposed deep cascade GRBF network will be lower than that of the AGRBF [28]. This is because the AGRBF has the same structure as the GRBF weak predictor. Specifically, the input to the AGRBF is $\boldsymbol{x}_t$, which is also the input to the cascaded network. But the input to the GRBF adaptive predictor in the cascaded network is $\boldsymbol{f}_t$. Through layerwise feature compression and extraction in the deep cascade network, the dimension of the input feature data $\boldsymbol{f}_t$ to the GRBF adaptive predictor is much smaller than the dimension of the original input data $\boldsymbol{x}_t$. Consequently, the online computational complexity of the GRBF adaptive predictor in the cascaded network will be smaller than that of the original AGRBF.

IV. Experimental Results

Three industrial applications, a debutanizer column process, a microwave heating process, and a penicillin fermentation process, are carried out to demonstrate the effectiveness of our deep cascade GRBF network.

Algorithm 1 Deep Cascade GRBF Network

```
1:  Initial training
2:  Construct $M_W$-term GRBF weak predictor based on training data set $\boldsymbol{D}_W = \{\boldsymbol{X}_{\text{tr}}, \boldsymbol{d}_{\text{tr}}; \boldsymbol{y}_{\text{tr}}\}$ using OLS algorithm.
3:  Calculate model output $\widehat{\boldsymbol{y}}_{\text{tr}}$ of trained weak predictor, and combine it with original training data to form training data set $\boldsymbol{D}_F = \{\boldsymbol{X}'_{\text{tr}}; \boldsymbol{y}_{\text{tr}}\}$ for SAE.
4:  Utilize $\boldsymbol{X}'_{\text{tr}}$ as input to progressively pretrain multiple AEs by minimizing cost (14) in unsupervised manner.
5:  Stack multiple AEs with an output regression layer, and fine-tune SAE based on $\boldsymbol{D}_F$ by minimizing cost (15).
6:  Obtain feature data $\boldsymbol{F}_{\text{tr}}$ from trained SAE, and form training data set $\boldsymbol{D}_A = \{\boldsymbol{F}_{\text{tr}}, \boldsymbol{d}_{\text{tr}}; \boldsymbol{y}_{\text{tr}}\}$.
7:  Construct initial $M_A$-term GRBF adaptive predictor based on $\boldsymbol{D}_A$ using OLS algorithm.
8:  Online prediction and adaptive modeling
9:  Set sample index $t = 1$.
10: Collect process's observations $\boldsymbol{x}_{p_t}$ of (17).
11: Form input $\boldsymbol{x}_t$ from $\boldsymbol{x}_{p_t}$, and calculate GRBF weak predictor's output $\widehat{y}_t = \boldsymbol{p}_{W_t}^{\text{T}} \boldsymbol{\theta}_W$.
12: Combine $\widehat{y}_t$ with $\boldsymbol{x}_{p_t}$ to form input $\boldsymbol{x}'_t$, and propagate $\boldsymbol{x}'_t$ through SAE to obtain deep output-relevant features $\boldsymbol{f}_t$.
13: With input $\boldsymbol{f}_t$, calculate GRBF adaptive predictor's output $\widetilde{y}_t = \boldsymbol{p}_{A_t}^{\text{T}} \boldsymbol{\theta}_{A_{t-1}}$, which is prediction of process output $y_t$.
14: When measurement of $y_t$ arrives, form $d_t = y_t - y_{t-1}$.
15: Calculate model performance $\widetilde{e}_t$ using (5) and (6).
16: IF $\widetilde{e}_t < \varepsilon$:
17:     Update GRBF adaptive predictor weight vector to $\boldsymbol{\theta}_{A_t}$ with RLS algorithm (7).
18: ELSE IF $\widetilde{e}_t \geq \varepsilon$:
19:     Calculate WNV values for all $M_A$ nodes using (8), and find $m = \arg\min_{1 \leq j \leq M_A} \text{WNV}_j$.
20:     Replace $m$th node with a new node by setting new center to $\boldsymbol{c}_m = \boldsymbol{x}_t$ and new scalar $\delta_m = d_t$.
21:     Calculate adaptive predictor's weight vector $\boldsymbol{\theta}_{A_t}$ as regularized LS estimate (10) and update $\boldsymbol{\Gamma}_t$ with (11).
22: END IF
23: Set $t = t + 1$ and go to line 10.
```

A. Experimental Setup

To measure the online prediction performance of an adaptive model, we consider the mean squared error (MSE)

$$\text{MSE}_t = \frac{1}{t} \sum_{i=1}^{t} (y_i - \widetilde{y}_i)^2 \tag{18}$$

and the mean absolute error (MAE)

$$\text{MAE}_t = \frac{1}{t} \sum_{i=1}^{t} |y_i - \widetilde{y}_i| \tag{19}$$

where $\widetilde{y}_i$ denotes the model prediction for the process output y_i. The online computational complexity of the adaptive model is quantified by the averaged computation time per sample (ACTpS). The computer for carrying out the experiments has the following configuration: Windows 10, 16 GB of RAM, CPU i7-9750 (2.60 GHz), and MATLAB version R2018b.

The proposed deep cascade GRBF network is compared with existing well-known online modeling approaches, including the RBF network [19], the GRBF network [28], which is also presented in Section II-A of this article, the TRBF network [28], and the AGRBF network [28], whose online prediction and adaptive modeling procedure is also presented in Section II-B of this article. In addition, two deep-learning models, the SAE [9], [10] and the LSTM [34], are also used for comparison. The original SAE [9], [10] is nonadaptive. We extend this SAE model to the adaptive SAE, and use it as the third deep-learning model benchmark. Explicitly, during the online operation, the adaptive SAE utilizes the RLS algorithm to update the weights of its linear regression layer.

The initial RBF, GRBF, TRBF, and AGRBF models are all constructed from the training data set with the OLS learning. During online operation, the RBF and GRBF networks only perform weight adaptation by the RLS algorithm, while the TRBF and the AGRBF adjust both weights and structure with the online prediction and adaptive learning procedures given in [24] and [28], respectively. For the SAE, multiple AEs are first pretrained in an unsupervised way layer by layer. After pretraining, a linear regression layer with single output neuron is added on the top of the stacked AEs for supervised fine-tuning. The entire SAE is trained by the stochastic gradient descent algorithm. For the LSTM with single hidden layer, Adam optimizer [39] is used to train the model based on the MSE cost. During online operation, the network structures and parameters of both the SAE and LSTM are fixed. The adaptive SAE has an identical training procedure to the SAE. During online operation, however, the adaptive SAE utilizes the RLS algorithm to update the weights of its linear regression layer.

The forgetting factor of the RLS algorithm is set to $\gamma = 0.98$ for all adaptive models. The node replacement of the TRBF involves a gradient descent optimization [24] whose step size and iteration number are empirically set to 0.1 and 5, respectively. The adaptive procedure of the AGRBF [28] is identical to the one for the GRBF adaptive predictor in the deep cascade network. The regularization parameter is set to $\lambda = 0.001$ for the regularized LS estimator (10). The important hyperparameters of all the models are chosen carefully by experiments, as detailed in the three case studies.

B. Case Study-I: Debutanizer Column Process

The debutanizer column [9], [34], [37] is an important unit of petroleum refinery industry, used to split desulfuration and naphtha. For process and product quality control, it is necessary to minimize butane content at the bottom of the column. The butane content measurement is normally obtained by the gas chromatography with large measurement delay. To deal with this problem, predictive model can be employed online to timely estimate the butane content. Seven process variables measured by sensors have good relevance with the quality variable, which can be used to construct the inferential model. Since there are strong nonlinearities and nonstationarity between the quality variable (output) and process variables (inputs), the proposed method with deep feature extraction capability is ideal for this online prediction and adaptive modeling task. Descriptions of the process inputs and output are given in Table I.

TABLE I
VARIABLE DESCRIPTION IN THE DEBUTANIZER COLUMN PROCESS

Process and quality variables		Description
Inputs	$u_{1,t}$	Top temperature
	$u_{2,t}$	Top pressure
	$u_{3,t}$	Reflux flow
	$u_{4,t}$	Flow to next process
	$u_{5,t}$	6th tray temperature
	$u_{6,t}$	Bottom temperature A
	$u_{7,t}$	Bottom temperature B
Output	y_t	Butane content

Based on the physiochemical insight and expert knowledge, the debutanizer column process can be represented by the following time-varying nonlinear system [9]:

$$y_t = f_{\text{sys}}\big(\boldsymbol{x}_{p_t}; t\big) \tag{20}$$

where $f_{\text{sys}}(\cdot; t)$ denotes the unknown time-varying nonlinear mapping of the system, and the input vector $\boldsymbol{x}_{p_t}$ is given by

$$\begin{aligned}\boldsymbol{x}_{p_t} = \big[&y_{t-1}\ y_{t-2}\ y_{t-3}\ y_{t-4}\ u_{1,t}\ u_{2,t}\cdots u_{5,t}\\ &\big(u_{6,t}+u_{7,t}\big)/2\ u_{5,t-1}\ u_{5,t-2}\ u_{5,t-3}\big]^T.\end{aligned} \tag{21}$$

Since the size of $\boldsymbol{x}_{p_t}$ is $n_p = 13$, the dimension of the input vector $\boldsymbol{x}_t$ to the GRBF weak predictor is 12, and the dimension of the input vector $\boldsymbol{x}'_t$ to the SAE unit in our deep model is 14. A total of 2390 samples of $\{\boldsymbol{x}_{p_t}; y_t\}$ are collected from the process, and the first 1000 samples are for training while the rest of them are for online prediction and adaptive modeling.

The sizes of the RBF, GRBF, TRBF, and AGRBF networks are all empirically chosen to be 10, as suggested in [28]. For a fair comparison, the size of the GRBF weak predictor in the proposed method is set to $M_W = 10$. For simplicity, we also set the size of the GRBF adaptive predictor to $M_A = 10$. The SAE unit in the proposed method consists of $n = 3$ AEs, and the neurons in the first, second, and third hidden layers are set to $\{10, 7, 4\}$, as suggested in [9]. With one output neuron at its regression output layer, the SAE unit has the network structure of $\{10, 7, 4, 1\}$. The training learning rate and maximum training epochs for the SAE unit are set to 0.01 and 200, respectively, as the training achieves convergence within 200 epochs. The node replacement threshold ε for the GRBF adaptive predictor is a vital hyperparameter, and we conduct a grid search over $\varepsilon \in \{1, 0.1, 0.01, 0.001, 0.0001, 0.00001\}$. The results obtained are depicted in Fig. 2. It can be seen that the best modeling accuracy is attained with $\varepsilon = 0.001$. Hence, we set $\varepsilon = 0.001$. Fig. 2 also indicates that reducing ε increases the ACTpS. This is expected, as smaller ε leads to more frequent node replacements, which, in turn, increases the computational complexity. The node replacement thresholds for the TRBF and AGRBF are also empirically set to 0.01 and 0.001, respectively. The SAE model has the same structure of the SAE unit in the proposed method. The adaptive SAE has the same network structure as the SAE model as well as the same training setting. For the LSTM, the hidden-layer size is determined by the grid search over the set of $\{16, 32, 64, 128, 256\}$, and the size of 32 is used as

TABLE II
PERFORMANCE COMPARISON OF RBF, GRBF, LSTM, SAE, ADAPTIVE SAE, TRBF, AGRBF, AND PROPOSED METHOD FOR DEBUTANIZER COLUMN PROCESS

Methods	Initial training		Online prediction/modeling		
	MSE (dB)	MAE	MSE (dB)	MAE	ACTpS (ms)
RBF	-38.8106	0.0081	-18.8683	0.0490	0.0067
GRBF	-41.5539	0.0062	-27.3516	0.0231	0.0077
TRBF	-38.8106	0.0081	-33.4602	0.0133	0.2015
AGRBF	-41.5539	0.0062	**-38.4860**	**0.0080**	**0.4989**
LSTM	-35.1764±1.1892	0.0134±0.0018	-30.3595±0.7991	0.0209±0.0024	NA
SAE	-51.3509±0.6323	0.0018±4.84e-5	-34.2239±4.4788	0.0092±0.0030	NA
Adaptive SAE	-51.0977±0.6402	0.0018±4.82e-5	-36.4189±4.0396	0.0084±0.0036	0.0021
Proposed	-42.4724±1.6812	0.0052±7.71e-4	**-40.5255±0.8252**	**0.0053±4.43e-4**	**0.2477**

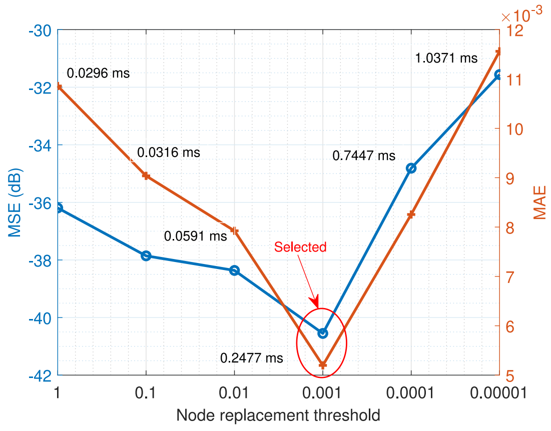


Fig. 2. Impact of node replacement threshold on the adaptive modeling accuracy and the online complexity (the black number [ms] denotes the ACTpS) of the proposed method for debutanizer column process.

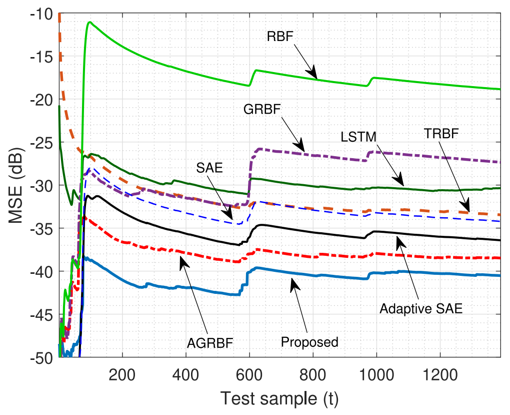


Fig. 3. MSE learning curves for online modeling of debutanizer column process by various models.

it attains the best performance. The learning rate and training epochs for the LSTM are empirically set to 0.001 and 200, respectively.

Both the training and online prediction performance attained by the eight models are compared in Table II, while Fig. 3 depicts the MSE learning curves for online prediction by these eight methods. It is well known that the performance of deep neural networks, such as the SAE, adaptive SAE, LSTM, and our deep cascade network, depend on initialization. Therefore, the average MSE and MAE over 20 independent experiments together with the corresponding standard deviations are listed in Table II for the SAE, adaptive SAE, LSTM, and the proposed method. The proposed method achieves much smaller standard deviations for the test MSE and MAE than those of the SAE and adaptive SAE. This means that our method is more robust than the SAE and adaptive SAE regarding initialization. Also, observe that the standard deviations of our method are similar to those of the LSTM.

Clearly, the training performance of both the RBF and TRBF is identical, while the training performance of the GRBF and AGRBF is the same, as training is carried out by the same OLS learning on the same training data. During online prediction, however, the TRBF is more than 14 dB better than the fixed-structure RBF while the AGRBF is more than 11 dB better than the fixed-structure GRBF. The online prediction performance of both the nonadaptive LSTM and SAE as well as the adaptive SAE degrade considerably from their respective training performance, particularly, the SAE type models, because their parameters and/or structures are fixed. This shows a serious drawback of applying deep neural networks, namely, inability to adapt their structures and parameters online. The adaptive SAE does attain a better performance when compared with the nonadaptive SAE, as it adapts its weights of the linear regression layer. Our proposed method outperforms all the other seven methods, attaining the smallest test MSE and MAE values. In particular, the online prediction MSE of our method is 2 dB lower than that of the second-best AGRBF. Also, observe that the online prediction MSE of the adaptive SAE is 4 dB higher than our method. This clearly demonstrates an important advantage of our proposed deep cascade GRBF network over the existing deep models, namely, our model is capable of adapting its structure online to track fast time-varying process characteristics.

In terms of online computational complexity, the RBF, GRBF, and adaptive SAE have the lowest ACTpSs, as they only update the models' linear weights using the RLS. By comparison, the ACTpSs of the TRBF, AGRBF, and our proposed deep cascade GRBF network are higher, because these models also adapt their networks' structures. But these three adaptive models all achieve very fast model structure updating at each sampling period, specifically, within a fraction of millisecond (ms). Therefore, even the process has a very high sampling rate, for instance, a sampling period of 1 ms, all the six adaptive models are capable of completing their adaption well within the sampling period constraint. Observe that the proposed deep cascade GRBF network imposes lower online complexity than the AGRBF. Specifically, the ACTpS of the AGRBF are 0.4989 ms, which is twice of the proposed method. The reason for this complexity reduction is explained in Section III-C.

To offer an intuitive comparison of the four adaptive models, the adaptive SAE, TRBF, AGRBF, and proposed method, their online predictions and errors are shown in Fig. 4. Observe that the error curve of the TRBF shows large fluctuations, indicating that it contains unmodeled parts. The AGRBF performs better but still has difficulty to predict some "turning points" (around 150, 600, and 1000 test samples). The error curve of the adaptive SAE also exhibits some unmodeled parts. The proposed method enables best tracking of this time-varying process with the smallest prediction errors.

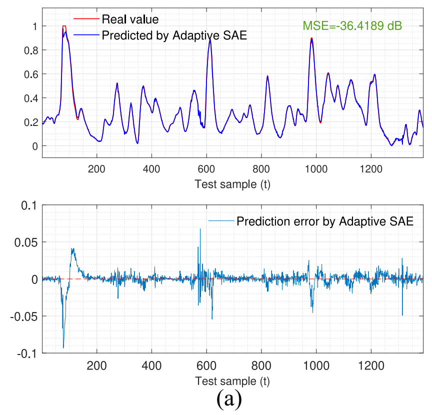
C. Case Study-II: Microwave Heating Process

Microwave energy, as a source of heat, has been widely used in industrial heating applications. However, due to the electromagnetic waves propagation properties and permittivity variation in materials, thermal runaway often occurs, leading to destructive results [40]. Therefore, it is crucial to predict the heating temperature online and detect thermal runaway in advance. Microwave heating process is time varying in nature, and the heating temperature is influenced not only by multi-physical field coupling but also by the permittivity variation in materials [41], [42]. This motivates us to investigate online adaptive model for real-time temperature prediction.

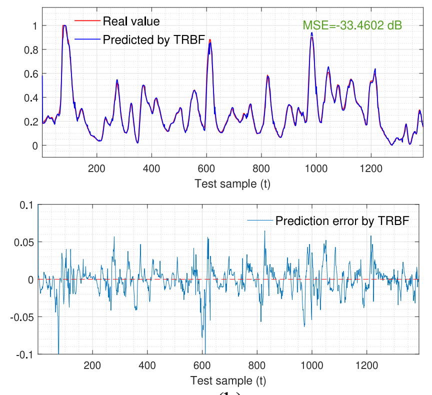
In the laboratory-scale microwave heating system [43], there are five microwave power sources of 3 kW each for a maximum power supply of 15 kW at 2.45 GHz. Microwave generated by each microwave source is transmitted through the corresponding waveguide, fed into multimode cavity, and finally absorbed by heated materials. The conveyor belt, whose speed is controlled by the motor driver, enables to continuously transport materials. Material temperature is measured by fiber optic sensors, which is sent to the host computer for monitor and control purpose. Five microwave powers and conveyor speed can be adjusted to control the temperature through the programmable logic controller (PLC) linked to the computer. Therefore, the process has six process input variables: the five microwave powers $u_{\mathrm{p}_i,t}$, $1 \leq i \leq 5$, and the conveyor speed $u_{\mathrm{v},t}$. The nonlinear time-varying relationship between the process inputs and the output, namely, the temperature y_t, can be described by the system $y_t = f_{\mathrm{sys}}(\boldsymbol{x}_{p_t}; t)$, with the system input vector defined by

$$\boldsymbol{x}_{p_t} = \left[y_{t-1}\ y_{t-2}\ y_{t-3}\ u_{\mathrm{p}_1,t-1} \cdots u_{\mathrm{p}_5,t-1}\ u_{\mathrm{v},t-1}\right]^T. \quad (22)$$

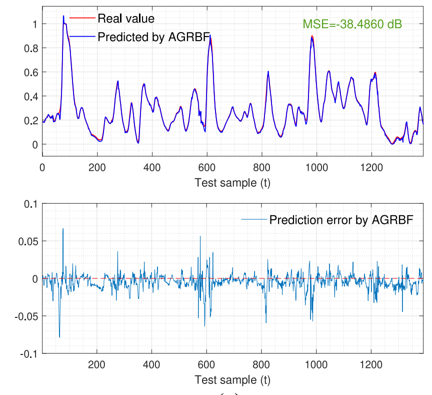
Since the dimension of $\boldsymbol{x}_{p_t}$ is $n_p = 9$, the dimension of the input vector $\boldsymbol{x}_t$ to the GRBF weak predictor is 8, and the dimension of the input vector $\boldsymbol{x}'_t$ to the SAE unit in our deep cascade network is 10. From this microwave heating system, 3000 samples $\{\boldsymbol{x}_{p_t}; y_t\}$ are collected in a real experiment [43]. We normalize the data into the range [0, 1], and separate them into the training (1000 samples) and online testing (2000 samples) sets.



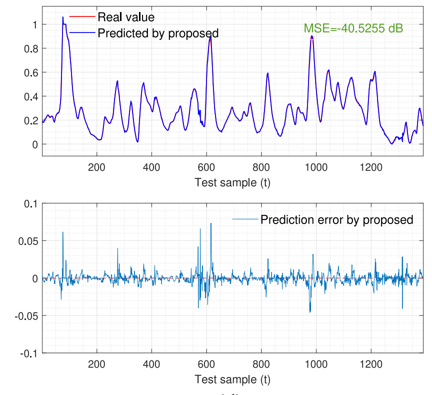
(a)



(b)



(c)



(d)

Fig. 4. Online prediction and error of: (a) adaptive SAE, (b) TRBF, (c) AGRBF, and (d) proposed method for debutanizer column process.

Again, the structures of all the models are carefully chosen by trial and error. Specifically, the numbers of nodes in the RBF, GRBF, TRBF, and AGRBF networks are all set to 10.

TABLE III
PERFORMANCE COMPARISON OF RBF, GRBF, LSTM, SAE, ADAPTIVE SAE, TRBF, AGRBF, AND PROPOSED METHOD FOR MICROWAVE HEATING PROCESS

Methods	Initial training		Online prediction/modeling		
	MSE (dB)	MAE	MSE (dB)	MAE	ACTpS (ms)
RBF	-33.0872	0.0166	-32.2692	0.0179	0.0054
GRBF	-30.1656	0.0210	-28.6259	0.0272	0.0057
TRBF	-33.0872	0.0166	-39.6183	0.0047	0.0320
AGRBF	-30.1656	0.0210	-41.7033	0.0041	0.0372
LSTM	-30.0209±2.0842	0.0256±0.0074	-30.3644±1.5311	0.0229±0.0066	NA
SAE	-42.4144±4.5252	0.0066±0.0034	-44.0824±4.1411	0.0054±0.0028	NA
Adaptive SAE	-40.6789±9.8211	0.0152±0.0224	**-44.2089±4.5475**	**0.0061±0.0065**	**0.0020**
Proposed	-43.9761±1.2653	0.0048±5.52e-4	**-46.0683±1.1564**	**0.0032±3.49e-5**	**0.0121**

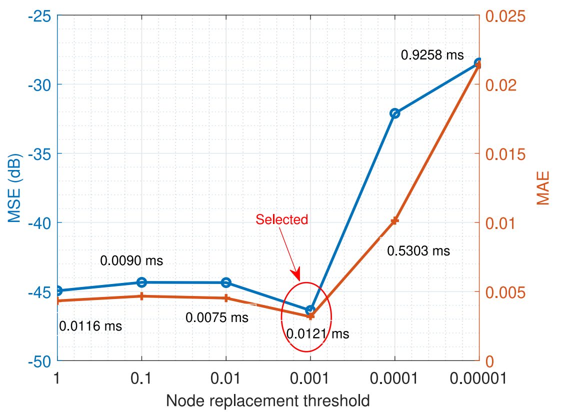


Fig. 5. Impact of node replacement threshold on the adaptive modeling accuracy and the online complexity (the black number [ms] denotes the ACTpS) of the proposed method for microwave heating process.

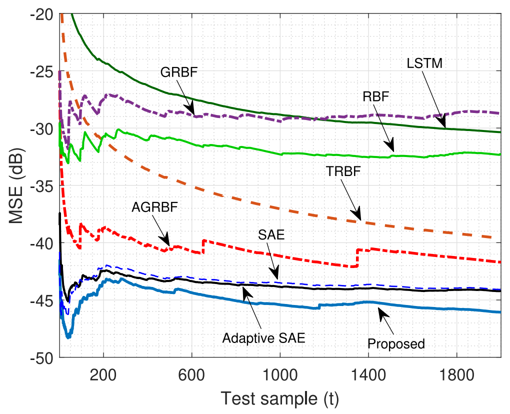


Fig. 6. MSE learning curves for online modeling of microwave heating process by various models.

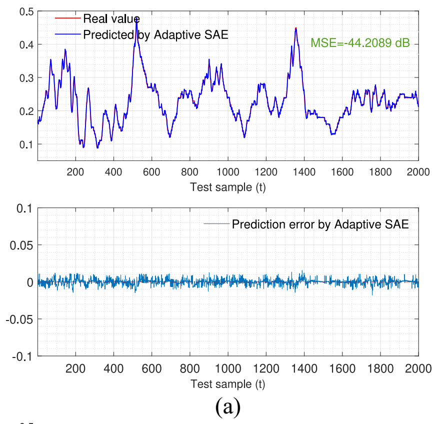
For our deep cascade network, we choose $M_W = M_A = 10$ for its weak and AGRBF predictors, while its SAE unit has the structure of $\{7, 5, 3\}$. The learning rate and the number of training epochs for the SAE unit are again set to 0.01 and 200, respectively. Based on the results of Fig. 5, we set the node replacement threshold for our proposed method to $\varepsilon = 0.001$. The LSTM network has 64 hidden nodes, and the structure of the SAE and adaptive SAE models are chosen to be $\{7, 5, 3, 1\}$. The learning rate and the number of training epochs are again set to 0.001 and 200, respectively, for the SAE, the adaptive SAE, and the LSTM. The node replacement thresholds are empirically chosen to be 0.1 and 0.01 for the TRBF and AGRBF, respectively.

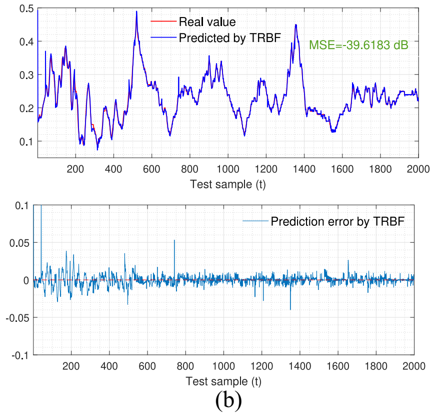
Table III lists both the training and online prediction performance achieved by the eight models, while Fig. 6 depicts the online MSE learning curves for the various models compared. Observe that both the SAE and adaptive SAE achieve very similar performance. Clearly, our method attains the best online prediction accuracy, and its online prediction MSE is 2 dB smaller than the second-best adaptive SAE. Moreover, the performance of the adaptive SAE exhibits a large fluctuation, due to its sensitivity to training initialization. Specifically, the standard deviation of the adaptive SAE's test MSE is 4.5475 dB, which is four times of our proposed method. In terms of online computational complexity, the adaptive SAE has the lowest ACTpS of 0.0020 ms among the three weight-adapting models, since it only updates three weights, while the proposed method attains the smallest ACTpS of 0.0121 ms among the three structure-adaptive models. Clearly, all the these six adaptive models are capable of meeting a very small sampling period constraint. The online predictions and errors of the four adaptive models, the adaptive SAE, TRBF, AGRBF, and proposed deep cascade network, are presented in Fig. 7, where it can be seen that the TRBF and AGRBF produce some large-magnitude online prediction errors. The adaptive SAE model has much better online prediction performance than the TRBF and AGRBF models, while the proposed method has the best online tracking accuracy.

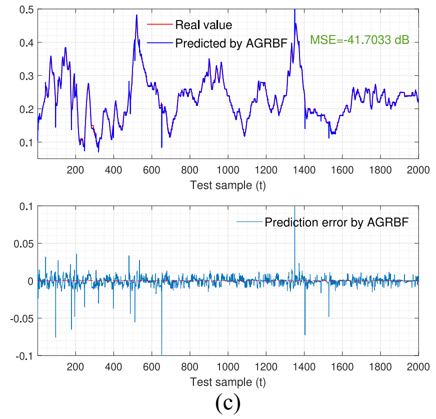
D. Case Study-III: Penicillin Fermentation Process

The penicillin fermentation process is an industrial biochemical fed-batch process with nonlinear dynamics and multimode characteristics, which has been widely adopted for performance assessment of adaptive soft sensors [44], [45]. The biomass concentration is a hard-to-measure key variable of this fermentation process, which is chosen as the system output, while other ten process variables are used as the system inputs, as tabulated in Table IV. This system can be represented by $y_t = f_{\text{sys}}(\boldsymbol{x}_{p_t}; t)$ [44], [45], with the system input vector specified by

$$\boldsymbol{x}_{p_t} = \left[u_{1,t}\ u_{2,t} \cdots u_{10,t}\right]^T. \quad (23)$$







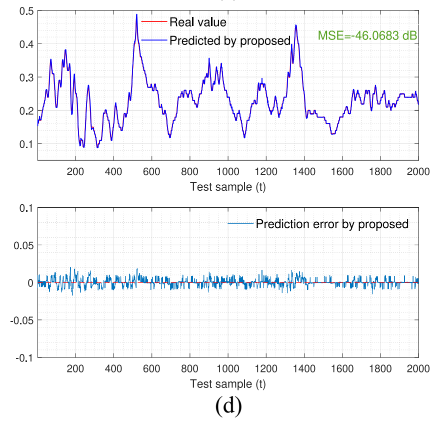


Fig. 7. Online prediction and error of: (a) adaptive SAE, (b) TRBF, (c) AGRBF, and (d) proposed method for microwave heating process.

Thus, the dimension of $\boldsymbol{x}_t$ is 10 and the dimension of $\boldsymbol{x}'_t$ is 11. For this fermentation process, 800 samples are collected from the PenSim tool [46] with default simulation condition. We separate them into the training (400 samples) and online testing (400 samples) sets.

TABLE IV
VARIABLE DESCRIPTION IN THE PENICILLIN FERMENTATION PROCESS

Process and quality variables		Description
Inputs	$u_{1,t}$	Aeration rate
	$u_{2,t}$	Agitator power
	$u_{3,t}$	Substrate feed rate
	$u_{4,t}$	Substrate feed temperature
	$u_{5,t}$	Dissolved oxygen concentration
	$u_{6,t}$	Culture volume
	$u_{7,t}$	Carbon dioxide concentration
	$u_{8,t}$	pH
	$u_{9,t}$	Fermentor temperature
	$u_{10,t}$	Generated heat
Output	y_t	Biomass concentration

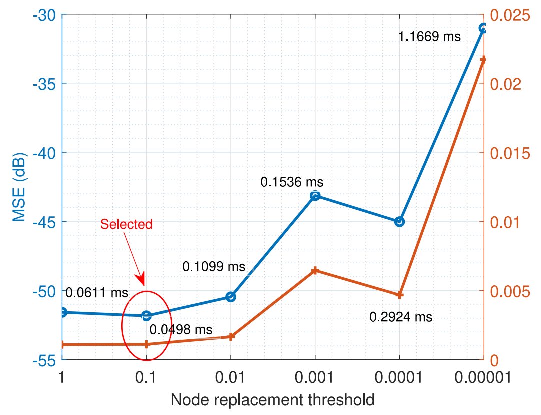


Fig. 8. Impact of node replacement threshold on the adaptive modeling accuracy and the online complexity (the black number [ms] denotes the ACTpS) of the proposed method for penicillin fermentation process.

Similarly, the structures of all the models are carefully chosen by experiments. Specifically, the numbers of nodes in the RBF, GRBF, TRBF, and AGRBF networks are all set to 10. For our deep cascade network, we choose $M_W = M_A = 10$ for its weak and AGRBF predictors, while its SAE unit has the structure of $\{10, 7, 4\}$. Based on the results of Fig. 8, we set the node replacement threshold to $\varepsilon = 0.1$ for the GRBF adaptive predictor in our deep cascade network. The LSTM network has 64 hidden nodes, and the structure of the SAE and adaptive SAE models are both chosen to be $\{10, 7, 4, 1\}$. The learning rate and and the number of training epochs for these three deep models are the same as the previous case. The node replacement thresholds are empirically chosen to be 0.1 and 1 for the TRBF and AGRBF models, respectively.

Table V lists both the training and online prediction performance of the eight models compared, while Fig. 9 plots the online MSE learning curves of these models. One may observe in Table V that the SAE and the adaptive SAE attain a spectacularly low training MSE performance but their online prediction accuracy degrade dramatically from the training performance. As expected, the adaptive SAE does attain the better test MSE than its nonadaptive counterpart. Again, our proposed deep method achieves the best online prediction accuracy, as evidenced by the smallest test MSE and MAE. The online prediction MSE of our method is 3 dB lower than

TABLE V
PERFORMANCE COMPARISON OF RBF, GRBF, LSTM, SAE, ADAPTIVE SAE, TRBF, AGRBF, AND PROPOSED METHOD FOR PENICILLIN FERMENTATION PROCESS

Methods	Initial training		Online prediction/modeling		
	MSE (dB)	MAE	MSE (dB)	MAE	ACTpS (ms)
RBF	-33.0446	0.0178	-23.5707	0.0548	0.0128
GRBF	-33.5760	0.0132	-22.9912	0.0520	0.0141
TRBF	-33.0446	0.0178	-31.6866	0.0141	0.1685
AGRBF	-33.5760	0.0132	-40.3049	0.0059	0.0600
LSTM	-32.6484±4.7905	0.0233±0.0138	-26.1942±4.4434	0.0507±0.0272	NA
SAE	-89.0346±9.7129	4.26e-5±3.72e-5	-45.9289±6.4825	0.0056±0.0041	NA
Adaptive SAE	-73.4173±10.2068	3.21e-4±5.46e-4	**-48.7861±5.2173**	**0.0022±0.0010**	**0.0033**
Proposed	-37.6025±1.1217	0.0139±7.75e-4	**–51.7963±1.9480**	**0.0012±1.09e-4**	**0.0498**

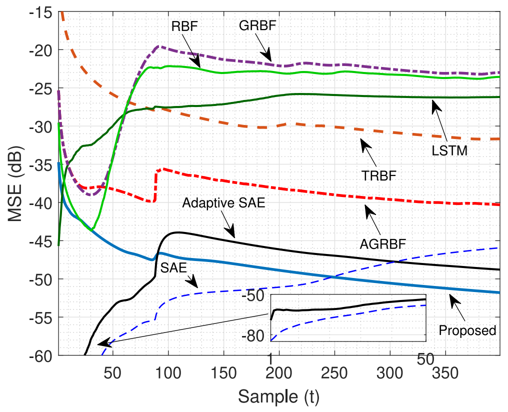


Fig. 9. MSE learning curves for online modeling of penicillin fermentation process by various models.

the second-best adaptive SAE. Also, observe that the deep cascade GRBF network imposes a lower online computational complexity than the AGRBF. Specifically, the online ACTpS of our deep model is less than 0.05 ms, compared to 0.06 ms imposed by the AGRBF. The online predictions and errors of the adaptive SAE, TRBF, AGRBF, and our proposed model are presented in Fig. 10, which again demonstrates the superior online tracking performance of our proposed method over the other models.

E. Discussion of the Results

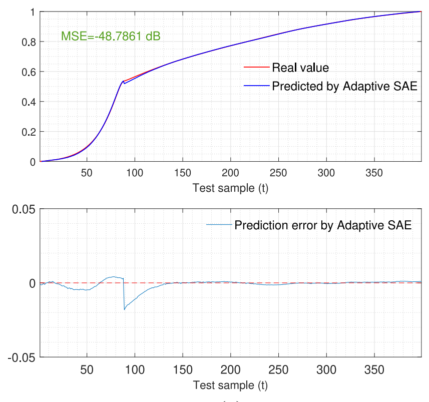
The experimental results involving three real-world industrial processes demonstrate that the proposed deep cascade GRBF network achieves the state-of-the-art adaptive modeling performance for high-dimensional nonlinear and nonstationary processes. Our proposed method not only consistently attains the best online prediction accuracy but also imposes sufficiently low online adaptation computational complexity that easily meets the constraint of small sampling period. In particular, compared with the current state-of-the-art deep neural network model, namely, the adaptive SAE, the online prediction MSE attained by our method is significantly lower than that of the adaptive SAE. Unlike our method, which can adapt its GRBF predictor's structure and weights online, the adaptive SAE can only adjust its output regression layer's weights, which is insufficient for tracking fast time-varying process's characteristics. The same limitation of the adaptive SAE, however, gives it an advantage in the online computational complexity, and the ACTpS of the adaptive SAE is significantly lower than that of our deep cascade GRBF network.

It can be seen that our proposed deep cascade GRBF network is particularly well designed for highly nonstationary data. If the underlying process is stationary or the process's dynamics only change slowly with time, our method may lose its competitive edge over the adaptive SAE. For high-dimensional slow time-varying nonlinear processes, the fixed SAE latent space is capable of extracting the compressed nonlinear features from raw data, and an adaptive linear regression layer becomes sufficient to track the slowly time-varying process characteristics. However, we emphasize again that this research is devoted specifically for severely time-varying data with nonlinear high dimensionality, and for such challenging application area, our deep cascade GRBF network shows considerable advantages over the existing state-of-the-art, as evidenced by the experimental results.

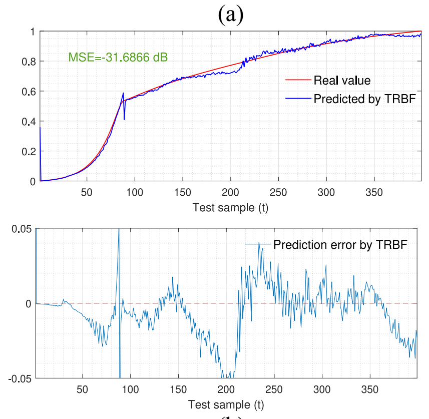
V. CONCLUSION

In this article, we have proposed a deep cascade GRBF network, consisting of a weak GRBF predictor, a SAE feature extractor, and a final GRBF adaptive predictor, aiming online prediction and modeling of nonlinear and nonstationary processes. The weak GRBF predictor provides a preliminary prediction result, which is combined with the raw input data to define the attribute input vector of a SAE. Deep output-relevant features are then extracted by the SAE, which provide the inputs of a GRBF adaptive predictor to improve modeling performance. Our deep cascade network seamlessly integrates deep learning with an AGRBF model. The proposed deep model with multilevel architecture can perform model adaptation very efficiently, enabling real-time tracking of quick changes in process dynamics within a small sampling period. Applications to three real-world industrial processes have validated the effectiveness of the proposed method over the existing state-of-the-art online adaptive modeling approaches and deep learning networks.

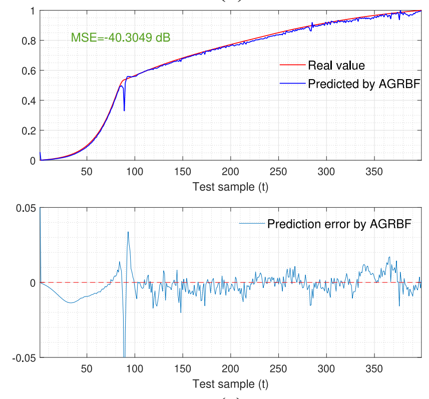
It is worth mentioning that supervised training of SAE with error backpropagation is prone to the local minimum problem,



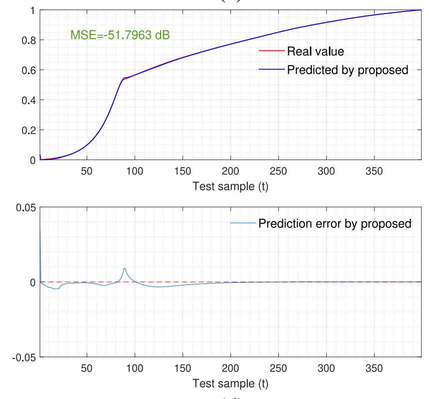
(a)



(b)



(c)



(d)

Fig. 10. Online prediction and error of: (a) adaptive SAE, (b) TRBF, (c) AGRBF, and (d) proposed method for penicillin fermentation process.

and therefore, the training is very sensitive to parameter initialization, which degrades the feature extraction robustness. If the training of the SAE unit can be made more robust by ensuring that the training procedure converges to a global or near global optimal solution, the nonlinear modeling capability of the entire deep cascade GRBF network can be further enhanced. How to construct a robust and an optimal SAE is an open research direction for a future works. A possible approach is adopting evolutionary optimization algorithms, such as differential evolutionary algorithm and swarm particle optimization, for supervised training.

REFERENCES

[1] C. Shang and F. You, "Data analytics and machine learning for smart process manufacturing: Recent advances and perspectives in the big data era," *Engineering*, vol. 5, no. 6, pp. 1010–1016, Dec. 2019.

[2] Z. Zhou, N. V. Chawla, Y. Jin, and G. J. Williams, "Big data opportunities and challenges: Discussions from data analytics perspectives," *IEEE Comput. Intell. Mag.*, vol. 9, no. 4, pp. 62–74, Nov. 2014.

[3] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in nonstationary environments: A survey," *IEEE Comput. Intell. Mag.*, vol. 10, no. 4, pp. 12–25, Nov. 2015.

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

[5] P. Zhou, W. Li, H. Wang, T. Chai, and M. Li, "Robust online sequential RVFLNs for data modeling of dynamic time-varying systems with application of an ironmaking blast furnace," *IEEE Trans. Cybern.*, vol. 50, no. 11, pp. 4783–4795, Nov. 2020.

[6] T. Liu, S. Chen, S. Liang, and C. J. Harris, "Selective ensemble of multiple local model learning for nonlinear and nonstationary systems," *Neurocomputing*, vol. 378, pp. 98–111, Feb. 2020.

[7] I. Škrjanc, J. A. Iglesias, A. Sanchis, D. Leite, E. Lughofer, and F. Gomide, "Evolving fuzzy and neuro-fuzzy approaches in clustering, regression, identification, and classification: A survey," *Inf. Sci.*, vol. 490, pp. 344–368, Jul. 2019.

[8] A. Seghouane and N. Shokouhi, "Adaptive learning for robust radial basis function networks," *IEEE Trans. Cybern.*, vol. 51, no. 5, pp. 2847–2856, May 2021.

[9] X. Yuan, Y. Wang, C. Yang, W. Gui, and B. Huang, "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.

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

[11] X. Yuan, Y. Wang, C. Yang, Z. Ge, Z. Song, and W. Gui, "Weighted linear dynamic system for feature representation and soft sensor application in nonlinear dynamic industrial processes," *IEEE Trans. Ind. Electron.*, vol. 65, no. 2, pp. 1508–1517, Feb. 2018.

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

[13] S. Chen, X. X. Wang, and C. J. Harris, "NARX-based nonlinear system identification using orthogonal least squares basis hunting," *IEEE Trans. Control Syst. Technol.*, vol. 16, no. 1, pp. 78–84, Jan. 2008.

[14] S. Chen, X. Hong, B. L. Luk, and C.J. Harris, "Orthogonal-least-squares regression: A unified approach for data modelling," *Neurocomputing*, vol. 72, nos. 10–12, pp. 2670–2681, Jun. 2009.

[15] S. Chen, X. Hong, and C. J. Harris, "Particle swarm optimization aided orthogonal forward regression for unified data modelling," *IEEE Trans. Evol. Comput.*, vol. 14, no. 4, pp. 477–499, Aug. 2010.

[16] X. Qian, H. Huang, X. Chen, and T. Huang, "Generalized hybrid constructive learning algorithm for multioutput RBF networks," *IEEE Trans. Cybern.*, vol. 47, no. 11, pp. 3634–3648, Nov. 2017.

[17] H. Han *et al.*, "Self-organizing RBF neural network using an adaptive gradient multiobjective particle swarm optimization," *IEEE Trans. Cybern.*, vol. 49, no. 1, pp. 69–82, Jan. 2019.

[18] S. Chen, S. A. Billings, and W. Luo, "Orthogonal least squares methods and their application to non-linear system identification," *Int. J. Control*, vol. 50, no. 5, pp. 1873–1896, 1989.

[19] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. Neural Netw.*, vol. 2, no. 2, pp. 302–309, Mar. 1991.

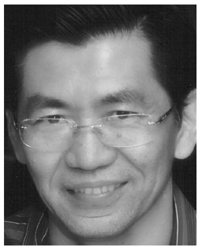
[20] X. Hong, S. Chen, J. Gao, and C. J. Harris, "Nonlinear identification using orthogonal forward regression with nested optimal regularization," *IEEE Trans. Cybern.*, vol. 45, no. 12, pp. 2925–2936, Dec. 2015.
[21] S. Chen, "Nonlinear time series modelling and prediction using Gaussian RBF networks with enhanced clustering and RLS learning," *Electron. Lett.*, vol. 31, no. 2, pp. 117–118, Jan. 1995.
[22] S. Chen and S. Billings, "Recursive prediction error parameter estimator for non-linear models," *Int. J. Control*, vol. 49, no. 2, pp. 569–594, 1989.
[23] N. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1411–1423, Nov. 2006.
[24] H. Chen, Y. Gong, X. Hong, and S. Chen, "A fast adaptive tunable RBF network for nonstationary systems," *IEEE Trans. Cybern.*, vol. 46, no. 12, pp. 2683–2692, Dec. 2016.
[25] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*, 5th ed. Hoboken, NJ, USA: Wiley, 2015.
[26] E. S. Chng, S. Chen, and B. Mulgrew, "Gradient radial basis function networks for nonlinear and nonstationary time series prediction," *IEEE Trans. Neural Netw.*, vol. 7, no. 1, pp. 190–194, Jan. 1996.
[27] T. Liu, S. Chen, S. Liang, S. Gan, and C. J. Harris, "Fast adaptive gradient RBF networks for online learning of nonstationary time series," *IEEE Trans. Signal Process.*, vol. 68, pp. 2015–2030, Mar. 2020.
[28] T. Liu, S. Chen, S. Liang, D. Du, and C. J. Harris, "Fast tunable gradient RBF networks for online modeling of nonlinear and nonstationary dynamic processes," *J. Process Control*, vol. 93, pp. 53–65, Sep. 2020.
[29] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.
[30] Y. Bengio, A. Courville, and P. Vincent, "Unsupervised feature learning and deep learning: A review and new perspectives," 2012, *arXiv:1206.5538v1*.
[31] X. Yuan, C. Yang, Y. Wang, B. Huang, J. Zhou, and W. Gui, "Hierarchical quality-relevant feature representation for soft sensor modeling: A novel deep learning strategy," *IEEE Trans. Ind. Informat.*, vol. 16, no. 6, pp. 3721–3730, Jun. 2020.
[32] C. Shang, F. Yang, D. Huang, and W. Lyu, "Data-driven soft sensor development based on deep learning technique," *J. Process Control*, vol. 24, no. 3, pp. 223–233, Mar. 2014.
[33] K. Wang, C. Shang, L. Liu, Y. Jiang, D. Huang, and F. Yang, "Dynamic soft sensor development based on convolutional neural networks," *Ind. Eng. Chem. Res.*, vol. 58, no. 26, pp. 11521–11531, May 2019.
[34] X. Yuan, L. Li, and Y. Wang, "Nonlinear dynamic soft sensor modeling with supervised long short-term memory network," *IEEE Trans. Ind. Informat.*, vol. 16, no. 5, pp. 3168–3175, May 2020.
[35] R. Xie, K. Hao, B. Huang, L. Chen, and X. Cai, "Data-driven modeling based on two-stream λ gated recurrent unit network with soft sensor application," *IEEE Trans. Ind. Electron.*, vol. 67, no. 8, pp. 7034–7043, Aug. 2020.
[36] C. Liu, L. Tang, and J. Liu, "A stacked autoencoder with sparse Bayesian regression for end-point prediction problems in steelmaking process," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 2, pp. 550–561, Apr. 2020.
[37] X. Yuan, Y. Gu, Y. Wang, C. Yang, and W. Gui, "A deep supervised learning framework for data-driven soft sensor modeling of industrial processes," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 11, pp. 4737–4746, Nov. 2020.
[38] X. Yuan, C. Ou, Y. Wang, C. Yang, and W. Gui, "A layer-wise data augmentation strategy for deep learning networks and its soft sensor application in an industrial hydrocracking process," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 8, pp. 3296–3305, Aug. 2021.
[39] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[40] S. Chandrasekaran, S. Ramanathan, and T. Basak "Microwave material processing—A review," *AIChE J.*, vol. 58, no. 2, pp. 330–363, 2012.
[41] B. Chen, B. Gao, C. Ge, and J. Li "Accurate solution and characteristics for electromagnetic wave propagation in time-varying media," *Modern Appl. Sci.*, vol. 3, no. 10, pp. 68–76, 2009.
[42] C. O. Kappe, "How to measure reaction temperature in microwave-heated transformations," *Chem. Soc. Rev.*, vol. 42, no. 12, pp. 4977–4990, 2013.
[43] K. Wang *et al.*, "Learning to detect local overheating of the high-power microwave heating process with deep learning," *IEEE Access*, vol. 6, pp. 10288–10296, 2018.
[44] W. Shao, S. Chen, and C. J. Harris, "Adaptive soft sensor development for multi-output industrial processes based on selective ensemble learning," *IEEE Access*, vol. 6, pp. 55628–55642, 2018.
[45] H. Jin, X. Chen, L. Wang, K. Yang, and L. Wu, "Dual learning-based online ensemble regression approach for adaptive soft sensor modeling of nonlinear time-varying processes," *Chemometr. Intell. Lab. Syst.*, vol. 151, pp. 228–244, Feb. 2016.
[46] "A Web Based Program for Dynamic Simulation of Fed-Batch Penicillin Production." [Online]. Available: http://simulator.iit.edu/web/pensim/simul.htm (accessed Jun. 2017).

Tong Liu (Member, IEEE) received the B.Sc. and Ph.D. degrees from the College of Automation, Chongqing University, Chongqing, China, in 2016 and 2021, respectively.

From 2018 to 2019, he visited the Next Generation Wireless Group, School of Electronics and Computer Science, University of Southampton, Southampton, U.K. He is currently working as a Research Associate with the OAK Group, Department of Computer Science, University of Sheffield, Sheffield, U.K. His research interests include computational intelligence, machine learning, data-driven modeling, and control of complex systems.

Zeyue Tian received the B.Sc. degree from the School of Mechanical Engineering, Beijing Institute of Technology, Beijing, China, in 2021. He is currently pursuing the master's degree with the School of Engineering, Hong Kong University of Science and Technology, Hong Kong, China.

From 2020 to 2021, he worked as a Research Assistant with the State Key Laboratory of Intelligent Technology and Systems, Department of Computer Science and Technology, Tsinghua University, Beijing. His research interests include machinery and electronics, robotics, and computer vision.



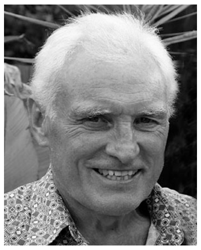
Sheng Chen (Fellow, IEEE) received the B.Eng. degree in control engineering from East China Petroleum Institute, Dongying, China, in 1982, the Ph.D. degree in control engineering from the City, University of London, London, U.K., in 1986, and the D.Sc. degree from the University of Southampton, Southampton, U.K., in 2005.

From 1986 to 1999, he held research and academic appointments with The University of Sheffield, Sheffield, U.K.; The University of Edinburgh, Edinburgh, U.K.; and University of Portsmouth, Portsmouth, U.K. Since 1999, he has been with the School of Electronics and Computer Science, University of Southampton, where he holds the post of a Professor of Intelligent Systems and Signal Processing. He has published over 650 research papers. He has over 17 200 Web of Science citations with H-index 57 and over 33 800 Google Scholar citations with H-index 79. His research interests include adaptive signal processing, wireless communications, modeling and identification of nonlinear systems, neural network and machine learning, and evolutionary computation methods and optimization.

Prof. Chen is a Fellow of the United Kingdom Royal Academy of Engineering, Asia–Pacific Artificial Intelligence Association, and IET. He is one of the original ISI highly cited researcher in engineering in March 2004.

Kai Wang received the B.S. degree in electronics and information engineering from Sichuan University, Chengdu, China, in 2000, and the Ph.D. degree in control theory and control engineering from Chongqing University, Chongqing, China, in 2009.

He was a Visiting Scholar with the School of Electrical and Computer Engineering, Cornell University, Ithaca, NY, USA, in 2014. He is currently an Associate Professor with the Department of Automation, Chongqing University. His current interests include machine learning, deep learning, and the application of big data.



Chris J. Harris received the B.Sc. degree from the University of Leicester, Leicester, U.K., in 1967, the M.A. degree from the University of Oxford, Oxford, U.K., in 1976, and the Ph.D. and D.Sc. degrees from the University of Southampton, Southampton, U.K., in 1972, and 2001, respectively.

He is an Emeritus Research Professor with the University of Southampton, having previously held senior academic appointments with Imperial College London, London, U.K.; University of Oxford, Oxford, U.K.; and The University of Manchester, Manchester, U.K., as well as the Deputy Chief Scientist for the U.K. Government. He is the coauthor of over 500 scientific research papers during a 50 year research career.

Prof. Harris was awarded the IEE Senior Achievement Medal for Data Fusion research and the IEE Faraday Medal for distinguished international research in machine learning. He was elected to Fellow of the U.K. Royal Academy of Engineering in 1996.