Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 8071810, 12 pages
https://doi.org/10.1155/2020/8071810

Research Article

A Novel Hybrid CNN-LSTM Scheme for Nitrogen Oxide Emission Prediction in FCC Unit

Wei He,[1,2] Jufeng Li,[2] Zhihe Tang,[2] Beng Wu,[1] Hui Luan,[2] Chong Chen,[1] and Huaqing Liang[1]

[1]*College of Information Science and Engineering, China University of Petroleum-Beijing, Beijing 102249, China*
[2]*HSE Testing Center, CNPC Research Institute of Safety and Environmental Technology, Beijing 102206, China*

Correspondence should be addressed to Chong Chen; chenchong@cup.edu.cn

Received 23 April 2020; Revised 22 June 2020; Accepted 28 July 2020; Published 17 August 2020

Guest Editor: Jun Chen

Fluid Catalytic Cracking (FCC), a key unit for secondary processing of heavy oil, is one of the main pollutant emissions of NO_x in refineries which can be harmful for the human health. Owing to its complex behaviour in reaction, product separation, and regeneration, it is difficult to accurately predict NO_x emission during FCC process. In this paper, a novel deep learning architecture formed by integrating Convolutional Neural Network (CNN) and Long Short-Term Memory Network (LSTM) for nitrogen oxide emission prediction is proposed and validated. CNN is used to extract features among multidimensional data. LSTM is employed to identify the relationships between different time steps. The data from the Distributed Control System (DCS) in one refinery was used to evaluate the performance of the proposed architecture. The results indicate the effectiveness of CNN-LSTM in handling multidimensional time series datasets with the RMSE of 23.7098, and the R^2 of 0.8237. Compared with previous methods (CNN and LSTM), CNN-LSTM overcomes the limitation of high-quality feature dependence and handles large amounts of high-dimensional data with better efficiency and accuracy. The proposed CNN-LSTM scheme would be a beneficial contribution to the accurate and stable prediction of irregular trends for NO_x emission from refining industry, providing more reliable information for NO_x risk assessment and management.

1. Introduction

Fluid Catalytic Cracking (FCC) is one of the most important technologies for secondary processing of heavy oil in refining and chemical enterprises [1]. Catalytic cracking reaction and catalyst regeneration are the main chemical processes of FCC. In the catalytic cracking reaction, crude oil is transformed into gasoline and diesel under catalysis during which 40%–50% of nitrogen in feedstock is transferred to coke and deposited on the catalyst [2–4]. Then, coke-covered spent catalysts are burned in the reaction regenerator for catalyst active regeneration, heat balance, energy recovery, and stable operation. During catalyst regeneration process, about 90% of the nitrogen in coke is converted into N_2 and the rest into NO_x and other reduced nitrogen compounds (NH_3, HCN, etc.). NO and NO_2 are the most detected NO_x which have potential risks to human health. As blood poison, NO would cause hemichypoxia and depress the central nervous system by strongly binding with hemoglobin (HB); NO_2 would cause bronchiectasis (even toxic pneumonia and pulmonary edema) by irritating and corroding the lung tissue [5, 6]. Furthermore, with the development of refining chemical technology, especially catalytic technique, more heavy oil with high percentage of nitrogen (such as residual oil and wax oil) were utilized. Therefore, it is urgent to accurately predict the NO_x produced during FCC process so as to effectively optimize the noxious gas discharged into the environment subject to the technical and economic conditions.

The FCC process is complex both from the modelling and from the control points of view [7–11]. Fortunately, many researchers have explored and developed semiempirical models, lumped kinetic models, and molecular-based kinetic models [12]. A comprehensive review on FCC

process modelling, simulation, and control was reported by [13]. Many research studies have been conducted using different models for modelling, controlling, and optimizing the FCC process with promising results [14–16]. With the development statistical learning theory, machine learning algorithms have proved effective methods for simulating natural systems in capturing nonlinearity with limited computation costs. The application of machine learning algorithms in the field of FCC is still at an early stage. Michalopoulos et al. [17] and Bollas et al. [18] proved the applicability of Artificial Neural Networks (ANN) in predicting the FCC products and optimized the operation conditions by developing ANN models for determining the steady-state behaviour of industrial FCC units. Zhang [19] established a NO_x emission model by Support Vector Machine (SVM) and further optimized the parameters with an improved adaptive genetic algorithm. Gu et al. [20] constructed a boiler combustion model on the basis of Least Support Vector Machines (LSSVM) and successfully forecasted NO_x emissions and other parameters which were verified by field data. Recent advantages in artificial intelligence (AI) (lead by deep learning) offered powerful predictive tool for effectively solving the highly complex chemical processes (such as FCC). Shao et al. proposed a new fault diagnosis method of chemical process by combining LSTM (Long Short-Term Memory) and CNN (Convolutional Neural Network) [21]. Yang et al. integrated deep neural network ("black box model") with lumped kinetic model (white box model) to create a novel "gray box model" for improving the efficiency and accuracy of simulating FCC process [22]. However, to the best of the authors' knowledge, there are few research studies using deep learning algorithms for predicting the NO_x emission in FCC unit. Some research studies of pollution emission problems have been conducted in power plants [23]. Compared to power plants, the FCC process is relatively complex with more factors involved. Therefore, it is of great difficulty to predict NO_x emissions in FCC units.

In this paper, a novel deep learning architecture for predicting NO_x emissions in the FCC Unit is proposed. The deep learning architecture is formed by integrating Convolutional Neural Network (CNN) and Long Short-Term Memory Network (LSTM) (refer as CNN-LSTM hereafter) with CNN layers extracting features among several variables and LSTM layers learning time series dependencies. The data from the Distributed Control System (DCS) in one refinery was used to demonstrate the performance of CNN-LSTM in the FCC unit. The main contributions of this paper are (1) the proposal of a novel hybrid CNN-LSTM scheme which is able to extract feature among different data sequences and the features between different time steps; (2) the application of the proposed scheme to predict NO_x emission during the FCC process with significant results.

2. Deep Learning Algorithms

2.1. Convolutional Neural Network Model (CNN). CNN is a special kind of neural network which is widely used in the field of image processing [24, 25]. In CNN, a feature map is used to extract the features from the input of the previous layer with a convolution operation. The pooling layer is used to reduce the computational complexity by reducing the size of the output from one stack layer to the next and at the mean time preserving important information. There are many pooling techniques available, among which maximum pooling is mainly used for pooling windows that contain maximum elements. The convolution layer provides the outputs of the pooling layer and maps it to the next layer. The last layer of CNN is usually fully connected for data classification. Figure 1 shows the basic architecture of CNN.

In neural network training, the accuracy and training speed could be affected by many factors [26]. For example, number of input layer nodes, number of hidden layers, number of hidden layer nodes, and the Internal Covariate Shift (ICS). That is to say, the inputs of the current layer would change according to the variation of parameters in the previous layers which would lead to more training time. In addition, if the inputs are distributed in ranges where the gradient of activation function is low, the ICS would cause the disappearance of gradient. In order to solve these problems, a Dropout method was included as follows.

Dropout (Figure 2) was first proposed by Hinton et al. in order to reduce the overfitting problem in neural networks [27–32]. In dropout procedure, the local feature dependency of the model will be reduced with a probability of P, and consequently, the generalization ability of the model will be improved effectively.

2.2. Long Short-Term Memory Network (LSTM). RNN is a kind of deep neural network which is specially used to process sequential data [33]. Compared with the traditional ANN, the characteristic of RNN is the inclusion of dependencies through time. The basic structure of a RNN is shown in Figure 3.

$$\alpha_t = b + Wh_{t-1} + Ux_t, \tag{1}$$

$$h_t = \tan(\alpha_t), \tag{2}$$

$$o_t = c + Vh_t, \tag{3}$$

$$\hat{y}_t = \text{soft max}(o_t). \tag{4}$$

The left side and the right side in the architecture are the folded form and the expanded form, respectively. In equations (1)~(4), t is time, x is the sequence of input data, h is the hidden layer state of the network, o is the output vector of the neuron, U is the parameter matrix from the input layer to the hidden layer, V is the parameter matrix from the hidden layer to the output layer, W is the parameter matrix between the hidden layers at different times, and $\hat{y}_t$ represents the probability output of the predicted value after normalization. All the parameter matrices are shared matrix of the hidden states at different times.

In order to solve the disappearance or explosion of gradient during training RNN, researchers proposed LSTM

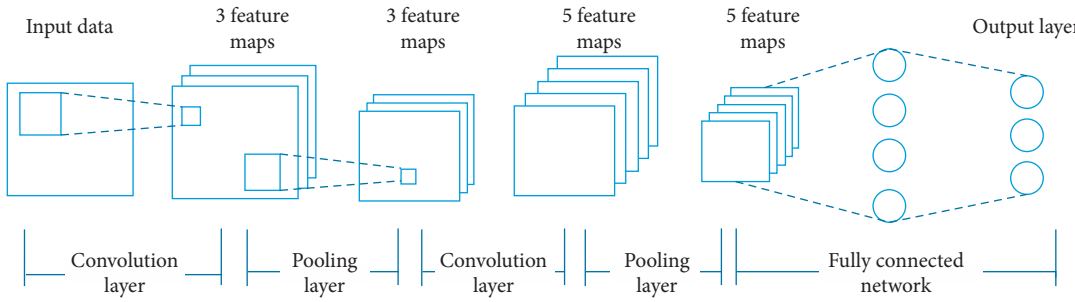


Figure 1: The architecture of the CNN.

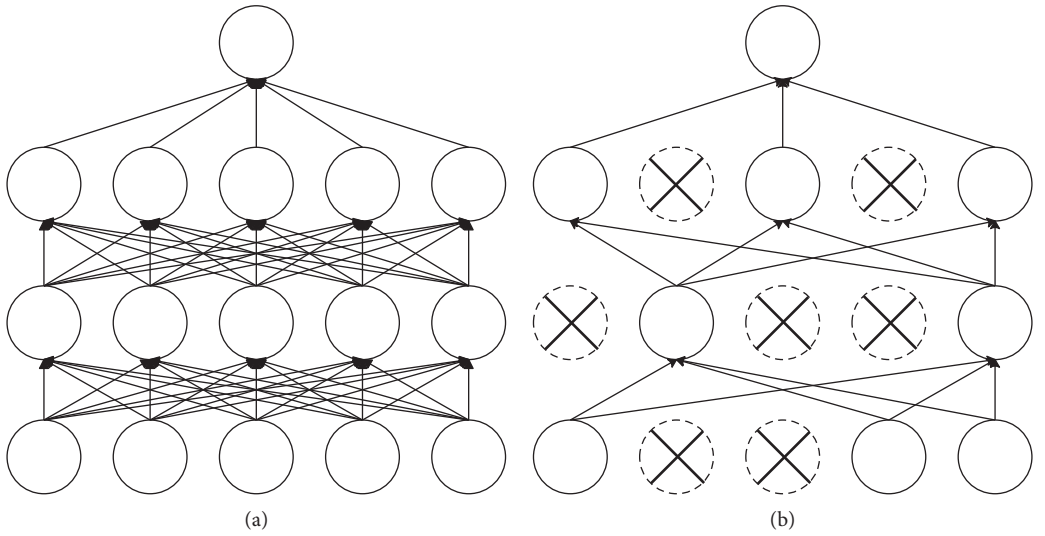


Figure 2: Dropout schematic: (a) Standard neural network; (b) after applying dropout.

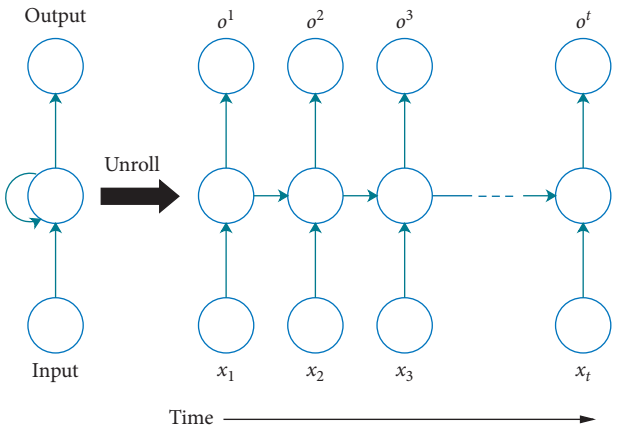


Figure 3: The architecture of the RNN.

by introducing gate mechanism in RNN [34, 35]. The gate mechanism is composed of input gate, output gate, and forgetting gate. As a special type of RNN, the neurons in the LSTM model are connected to each other in a directed cycle. The basic structure of LSTM is shown in Figure 4.

The LSTM model saves long-term dependencies using three different gates in an effective way. The structure of LSTM (shown in Figure 4) is similar to RNN. LSTM uses three gates to regulate and preserve information into every node state. The explanation of LSTM gates and cells is provided in equations (5)~(8):

$$\text{Input Gate } \text{In}_t = \sigma\left(W_{\text{in}} \cdot \left[h_{st} - 1\right], x_t + b_{\text{in}}\right), \tag{5}$$

$$\text{Memory Cell } C_t = \tanh\left(W_c \cdot \left[h_{st} - 1\right], x_t + b_e\right), \tag{6}$$

$$\text{Forget Gate } f_t = \sigma\left(W_f \cdot \left[h_{st} - 1\right], x_t + b_f\right), \tag{7}$$

$$\text{Output Gate } O_t = \sigma\left(W^{\circ} \cdot \left[h_{st} - 1\right], x_t + b^{\circ}\right), \tag{8}$$

where b represents the bias vector; W is weight matrix; x_t is the input vector at time t; and In, f, C, and O represent input, forget, cell memory, and output gates, respectively.

3. CNN-LSTM

Due to the characteristics of CNN and LSTM, a common thought to combine the advantages is to integrate CNN and LSTM. In this study, a new deep learning scheme was proposed by integrating CNN and LSTM. Two layers of CNN were used to ensure the correlation and effective extraction of multidimensional data. The feature sequence from the CNN layer was considered as the input for LSTM.

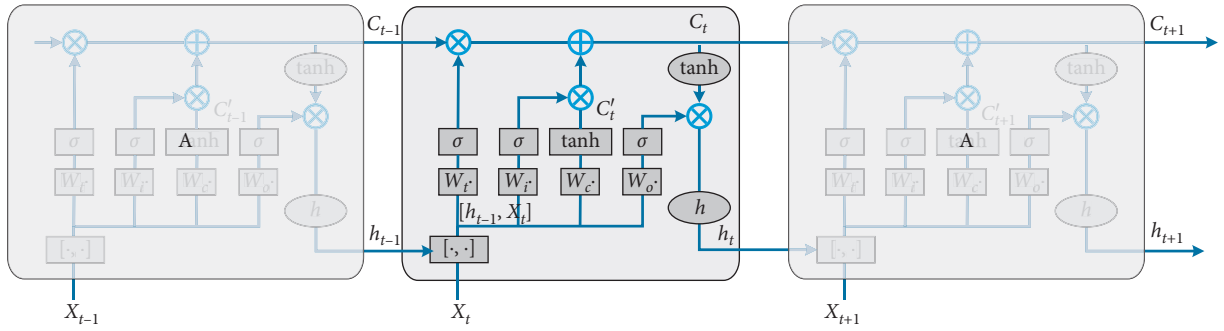


Figure 4: The architecture of the LSTM.

The time dependencies were further extracted in the LSTM layer. Three fully connected layers existed in the architecture which refer to FC1, FC2, and FC3. FC1 and FC2 are used to obtain the features extracted by the CNN layer, and FC3 is used to conduct the final data prediction. Figure 5 shows the architecture of the proposed CNN-LSTM.

3.1. CNN Layer. The input data (train_x) and output data (train_y) are defined as follows:

$$\text{train\_}x_i = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1q} \\ x_{21} & x_{22} & \cdots & x_{2q} \\ \cdots & \cdots & \cdots & \cdots \\ x_{p1} & x_{p2} & \cdots & x_{pq} \end{bmatrix},$$

$$\text{train\_}y_i = \left[\text{var}_{t-p+3}, \text{var}_{t-p+4}, \ldots, \text{var}_{t+1}, \text{var}_{t+2}, \ldots, \text{var}_{t+q},\right]^T, \tag{9}$$

where p represents time step and q represents data features.

The ith sample from the training set is fed into the network. In the first convolution layer (1stConV), the convolution kernel size, number, and step length are denoted as filter_size = (m, n), *filter_num*, and *strides*, respectively.

The jth convolution kernel W_j is defined as follows:

$$W_j = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ w_{m1} & w_{m2} & \cdots & w_{mn} \end{bmatrix}. \tag{10}$$

The algorithms between jth convolution kernel W_j and input train_x_i could be described as follows:

$$W_j \odot \text{train\_}x_i = \text{featureMap\_}x_i = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1b} \\ x_{21} & x_{22} & \cdots & x_{2b} \\ \cdots & \cdots & \cdots & \cdots \\ x_{a1} & x_{a2} & \cdots & x_{ab} \end{bmatrix}. \tag{11}$$

The operation for convolution layer is denoted as $\odot$, where

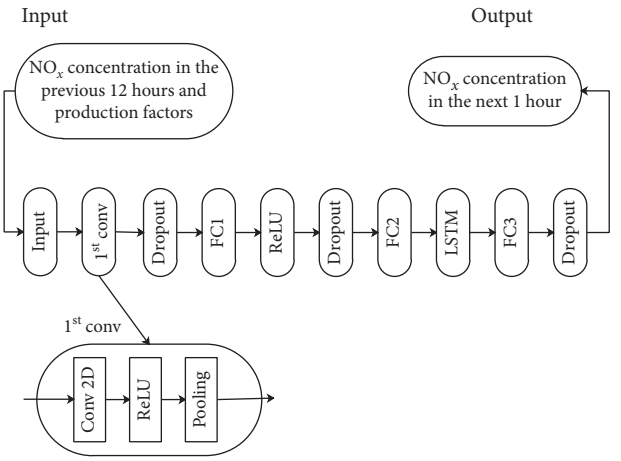


Figure 5: Architecture of the proposed CNN-LSTM.

$$a = \frac{p - m + 2 \times \text{padding}}{\text{strides}} + 1,$$
$$b = \frac{q - n + 2 \times \text{padding}}{\text{strides}} + 1. \tag{12}$$

The element x in the feature map is obtained through multiplying W_j by Receptive Field, which is recorded as follows:

$$\text{train\_}x_i\text{\_field} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}. \tag{13}$$

$x \in \text{featureMap\_}x_i = W_j \bigcirc \text{train\_}x_i\text{\_field} = \sum_{k=1}^{m} \sum_{l=1}^{n} w_{kl}\, x_{kl}$, where $\bigcirc$ means multiply the elements.

1$_{st}$ConV is calculated as

$$\begin{aligned} 1^{st}\text{ConV}_{x_i} &= W \odot \text{train}_{x_i} \\ &= \left[\text{featureMap\_}x_1, \text{featureMap\_}x_2, \ldots, \right. \\ &\quad \left. \text{featureMap\_}x_k\right], \end{aligned} \tag{14}$$

where $W = [W_1, W_2, \ldots, W_k]$

ReLU is used as the activation function:

$$f_{\mathrm{ReLU}} = \begin{cases} x, & x > 0, \\ 0, & x \le 0. \end{cases} \tag{15}$$

The output of the convolutional layer is nonlinear mapping by the activation function. In pooling layer, the data are compressed and recorded as pooling_size = (m', n').

For every feature map,

$$x_{i\_\mathrm{pool}} = f_{\mathrm{pool}}(\mathrm{featureMap}\_x_i) = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1b_1} \\ x_{21} & x_{22} & \cdots & x_{2b_1} \\ \cdots & \cdots & \cdots & \cdots \\ x_{a_1 1} & x_{a_1 2} & \cdots & x_{a_1 b_1} \end{bmatrix}, \tag{16}$$

where $a' = a/m', b' = b/n'$

Thus, the ith sample after convolutional, activation, and pooling layer is

$$X_{\mathrm{pool}\_1} = \left(x_{1\_\mathrm{pool}}, x_{2\_\mathrm{pool}}, \ldots, x_{k\_\mathrm{pool}}\right). \tag{17}$$

The convolutional, activation, and pooling in 2ndConV are similar to those in 1stConV.

Dropout is denoted as dropout (λ); λ takes the value between 0 and 1, which means the percentage of the data that should be discarded. For instance, dropout (0.5) means that 50% of neuron data are discarded randomly.

FC layer dense (α) is the output data in the last dimension. For the above input type [none, a', b', k], only the last dimension [none, a', b', α] is changed after full connection.

Transform [samples, height, width, channels] to [samples, timesteps, features], and then feed them in the LSTM layer. The modular construction of LSTM is shown as follows, in which forget, input, and output gates are included.

3.2. LSTM Layer. The forget gate is expressed as follows:

$$f_t = \sigma\left(W_f \cdot \left[h_{t-1}, x_t\right] + b_f\right), \tag{18}$$

where W_f represents the weight matrix for the forget gate; $[h_{t-1}, x_t]$ means concatenation of h_{t-1} and x_t; b_f represents the offset of the forget gate; and σ represents the sigmoid function. The dimensionality of input layer, hidden layer, and cell state is d_x, d_h, and d_c, respectively. In general, $d_c = d_h$, the dimensionality of weight matrix for the forget gate, and W_f is $d_c \times (d_h + d_x)$. Actually, the weight matrix (W_f) is combined by two matrices W_{fh} (initem: h_{t-1}; dimensionality: $d_c \times d_h$) and W_{fx} (initem: x_t; dimensionality: $d_c \times d_x$), W_f could be written as follows:

$$\begin{aligned} \left[W_f\right]\begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} &= \begin{bmatrix} W_{fh} & W_{fx} \end{bmatrix}\begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} \\ &= W_{fh}h_{t-1} + W_{fx}x_t + W_{fx}x_t. \end{aligned} \tag{19}$$

Input gate could be expressed as follows:

$$i_t = \sigma\left(W_i \cdot \left[h_{t-1}, x_t\right] + b_i\right), \tag{20}$$

where W_i represents the weight matrix for the forget gate and b_i represents the offset of the input gate.

The cell state for input description is calculated by the last output data and the current input data:

$$\widetilde{c}_t = \tanh\left(W_c \cdot \left[h_{t-1}, x_t\right] + b_c\right). \tag{21}$$

The current cell state (C_t) is as follows:

$$C_t = f_t \circ C_{t-1} + i_t \circ \widetilde{C}_t, \tag{22}$$

where the last cell state (C_{t-1}) is multiplied by forget gate (f_t) according to different element and the current input cell state ($\widetilde{C}_t$) is multiplied by input gate (it) according to different element.

The new cell state (C_t) is established by current memory ($\widetilde{C}_t$) and long-term memory C_{t-1}. On one hand, due to the mechanism of forget and input gate, the new cell state store information from a long time ago or forget the irrelevant content. On the other hand, the output gate controls the effect of long-term memory on current output:

$$o_t = \sigma\left(W_o \cdot \left[h_{t-1}, x_t + b_o\right]\right). \tag{23}$$

The final output of LSTM is decided by the output gate and cell state (equation (29):

$$h_t = o_t \circ \tanh c_t. \tag{24}$$

3.3. Realization of CNN-LSTM. The CNN-LSTM was realized in Keras using TensorFlow backend based on Figure 5 and the theory described in the previous sections (shown in Algorithm 1). After normalization, the training data (train_x, train_y) was fed into the constructed CNN model (1st ConV_model) to train the parameters with loss function (loss_function which is "mae" in our case) and optimizer (optimizer, which is "adam" in our case). The feature map of CNN was then extracted and reshaped to train the LSTM layer.

4. Experiments

4.1. Datasets. Several key production factors that affect the nitrogen oxide concentration in the plant were selected from 276 kinds of production factors of catalytic cracking unit. By inquiring experts, the key factors of production include nitrogen content in raw materials, process control parameters of reactor (FCC reaction temperature, catalyst/oil ratio, and residence time), the regeneration process control parameters (regeneration way, dense bed temperature, oxygen content in furnace, and carbon monoxide concentration), and catalyst species (platinum CO combustion catalyst and nonplatinum CO combustion catalyst).

A total of 2.592×10^5 of samples collected in half a year were divided into training and validation sets with the proportion of 70% and 30%, respectively. As shown in Table 1, the key production factors were used as input data and the NO_x emission were used as labels.

In order to eliminate the dimensional effects among different variables, the original data was standardized using the MinMaxScaler function in Python (equations (25) and (26)):

Input: train_x, train_y
Hyper-parameters: filters, kernel_size, pool_size, batch_size, rate
Initialize ()
Normalization (train_*x*, train_*y*)
//The first convolution layer
1st ConV_model = Sequential ([Convolution2D (filters, kernel_size, name = "Conv2D_1"), MaxPooling2D (pool_size), Flatten (), Dense (units, activation), Dropout (rate), Dense (units, activation)])
1st ConV_model.compile (loss_function, optimizer)
1st ConV_model.fit (train_*x*, train_*y*, epochs, batch_size)
//Extract the feature map
1st ConV_feature_model = Model (inputs, 1st ConV_model.get_layer ("Conv2D_1").output)
1st ConV_feature_output = 1st ConV_feature_model.predict (train_*x*)
//LSTM layer
reshape (1st ConV_feature_output)
LSTM_model = Sequential (LSTM (units, activation, recurrent_activation), Dense (units, activation))
LSTM_model.compile (loss_function, optimizer)
LSTM_model.fit (1st ConV_feature_output, train_*y*, epochs, batch_size)

Algorithm 1: Pseudocode of CNN-LSTM.

$$X_{\text{std}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}, \tag{25}$$

$$X_{\text{scaled}} = \frac{X_{\text{std}}}{(\max - \min) + \min}, \tag{26}$$

where $X_{\max}$ and $X_{\min}$ are the maximum and minimum values of the data and max and min are maximum and minimum values of the zoom range. In addition, the problem of time prediction was reconstructed into supervised learning.

4.2. Hyperparameters

4.2.1. CNN. The hyperparameters in RNN mainly contain weight initialization, learning rate, activation function, epoch numbers, iteration times, etc. Several important hyperparameters include number of convolution layers, number of convolution kernels, and size of convolution kernel are discussed in this study.

4.2.2. LSTM. Long-short term memory (LSTM) is a kind of RNN, in which tanh could be replaced by sigmoid activation function, resulting in faster training speed. In LSTM, Adam was used as an optimizer, MSE was used as a loss function, and identity activation function was used to complete the weight initialization. The hyperparameters in LSTM mainly contain number of hidden layer nodes and the number of batch sizes. The number of hidden layer nodes in LSTM have direct influences on the learning results by affecting the ability of nonlinear mapping which is the same as in Feedforward Neural Networks. The batch size have an influence on the computation costs and the learning accuracy by affecting amount of data used for updating the gradient.

4.2.3. CNN-LSTM. As a neural network model combined CNN with LSTM, the hyperparameters of CNN-LSTM is basically the same with CNN and LSTM which mainly include learning rate η, regularization parameter λ, the number of neurons in each hidden layer (such as the full-connected layer and the number of neurons in LSTM), batch size, convolution kernel size, neuron activation function, pool layer size, and dropout rate. All the related hyperparameters were investigated and analysed in Section 5.

4.3. Performance Criteria. The performances of different algorithms were evaluated by the Root Mean Square Error (RMSE) (equation (27) and the coefficient of determination (R^2) (equation (28)) [36]. The RMSE value reflects the discrete relationship between predicted and observed values:

$$\text{RMSE} = \sqrt{\frac{\sum_{n=1}^{N} (o_n - p_n)^2}{N}}, \tag{27}$$

where N is the data length, o_n is the nth observed value, and p_n is the nth predicted value.

The R^2 value reflects the accuracy of the model which ranges from 0 to 1 with 1 denotes perfect match:

$$R^2 = \frac{\text{SSR}}{\text{SST}} = \frac{\sum_{i=1}^{n} (\hat{y}_i - \overline{y})^2}{\sum_{i=1}^{n} (y_i - \overline{y})^2}, \tag{28}$$

where $\hat{y}_i$ represents the predicted value, $\overline{y}$ is the average value, and y_i is the observed value.

5. Results and Discussion

5.1. CNN-LSTM. The hyperparameters mentioned above were determined by the trial-and-error method. RMSE and R^2 were considered as objective function to optimize the size and number of convolution kernel, the number of batch size, the number of convolution layers, and the probability of dropout. The results shown in Figure 6 indicate the process of optimizing hyperparameters for the proposed method.

Table 1: The setting of input and labels.

Data types	Key production factor types	Parameters	Unit
Input data	Raw materials	Nitrogen content	%
	Process control parameters of reactor	FCC reaction temperature	˚C
		Catalyst/oil ratio	%
		Residence time	Second (s)
	Regeneration process control parameters	Regeneration way	
		Dense bed temperature	˚C
		Oxygen content in furnace	%
		Carbon monoxide concentration	% mg/m^3
	Catalyst species	Platinum CO combustion catalyst	—
		Nonplatinum CO combustion catalyst	
Labels	NO_x emission data		mg/m^3

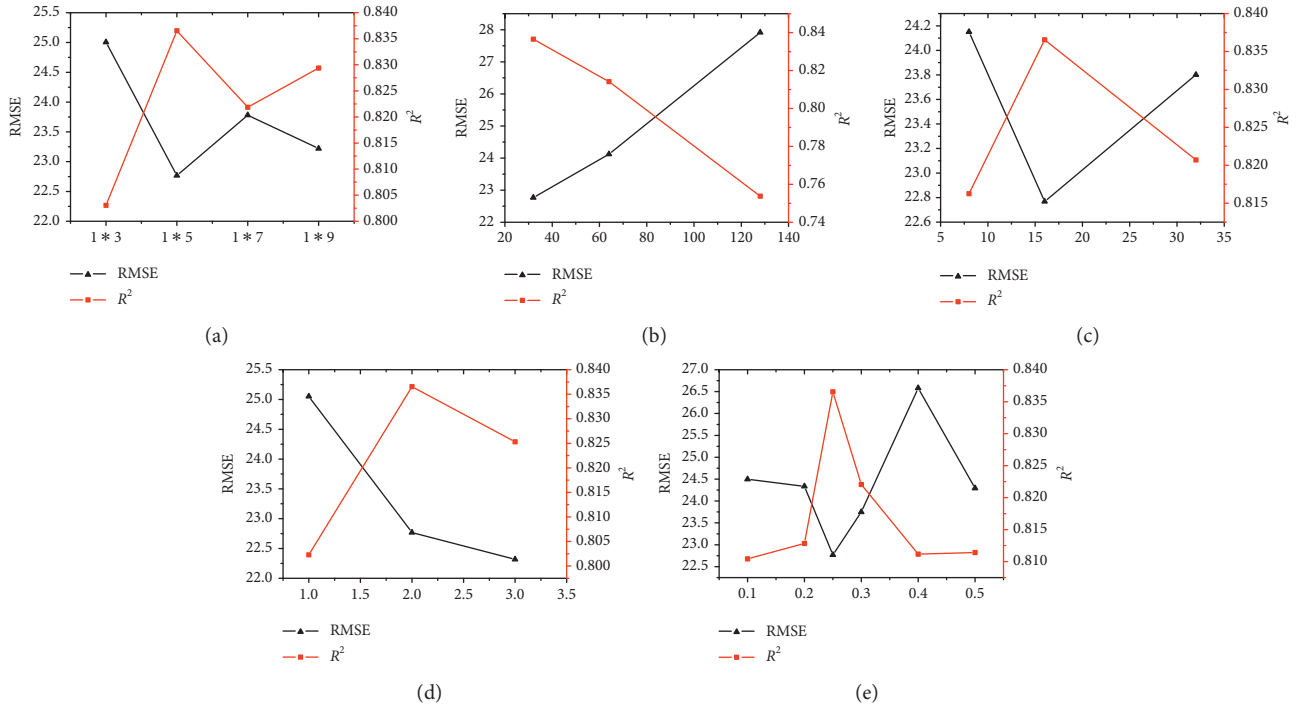


Figure 6: Optimization of hyperparameters for CNN-LSTM. (a) Size of convolution kernels. (b) The number of batch size. (c) The number of convolution kernels. (d) The number of convolution layers. (e) The probability of dropout.

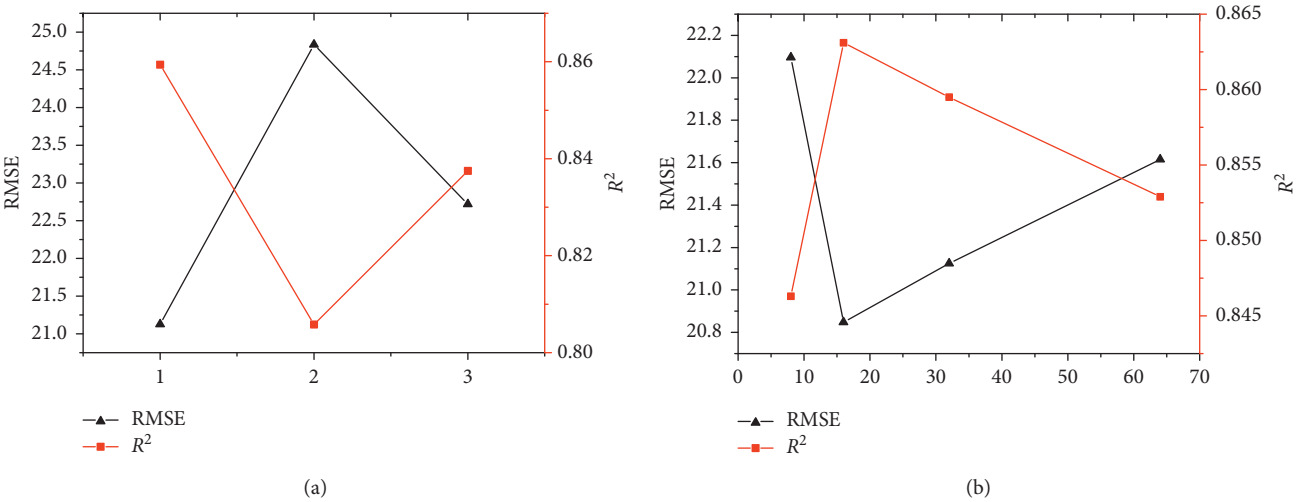
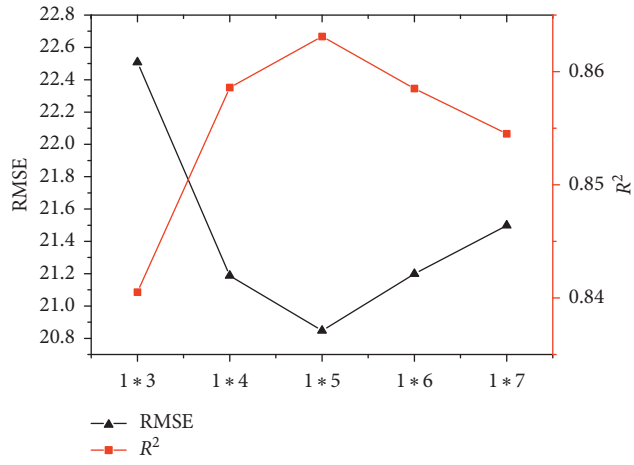


Figure 7: Continued.



(c)

Figure 7: Optimization process of hyperparameters for CNN. (a) The number of convolution layers. (b) The number of convolution kernels. (c) Size of convolution kernel.

The network structure adopts two convolutional parts as the CNN layer; the kernel size is 1×5 for the first and second CNN layer. Each convolution layer is followed by a Rectified Linear Unit (ReLU) layer (equation (29) and a maximum pooling layer. The output of CNN part is a 32-dimensional vector after operations. All the vectors form a sequence and feed into the LSTM layer:

$$a_{i,j,k} = \max\left(z_{i,j,k}, 0\right), \tag{29}$$

where $z_{i,j,k}$ is the input of the activation function at location (i, j) on the kth channel. ReLU allows neural networks to compute faster than sigmoid or tanh activation functions and train deep network more effectively. In order to train a neural network with strict backpropagation algorithm, the contribution of all samples to the gradient must be considered simultaneously.

With the incorporation of the LSTM network, the proposed CNN-LSTM network can be trained with time series data of FCC unit. A LSTM layer followed by the FC layer is used to assign the predicted value to each frame in the sequence.

The output of the CNN layer passes through two dropout layers and two FC layers to combine the features extracted by the CNN layer. During the training stage, the dropout layer will randomly remove the connection between the CNN layer and the FC layer in each iteration. In our experiments, we set the dropout rate to an empirical value of 0.25, which has shown effectiveness in performance improvement (the experiment on the dropout rate is shown in Figure 6(e)). The convolution layer, convergence layer, and activation function layer are conducted to map the raw data to the feature space in hidden layer. And the full-connected layer plays the role of "classifier" which maps the learned feature representation to the memory space of the sample.

5.2. CNN. The hyperparameters of CNN were also determined by the trial-and-error method. RMSE and R^2 were considered as an objective function to optimize the size and number of convolution kernel and the number of convolution layers. The results shown in Figure 7 indicate the process of optimizing hyperparameters for CNN from which one can conclude that the optimal values for the number of convolution layers, the number of convolution kernels and the size of convolution kernel are 1, 16, and 1×5.

5.3. LSTM. The optimization process of hyperparameters for LSTM was shown in Figure 8. RMSE and R^2 were used as quantitative performance criteria to evaluate the hyperparameters (i.e., the number of hidden layer nodes and the number of batch size). The process and results shown in Figure 8 indicated the optimal values for the number of hidden layer nodes and the number of batch size are 40 and 500, respectively.

5.4. Experiments on Different Methods. The accuracy of CNN and LSTM and the proposed CNN-LSTM method for the training stage and validation stage were evaluated by R^2 and RMSE. All methods were well tuned and ten test runs were conducted to eliminate the random errors of each method. The average criteria for each method were calculated to evaluate the performance. The results were presented in Tables 2 and 3 and Figure 9, respectively. Compared with traditional CNN, the proposed CNN-

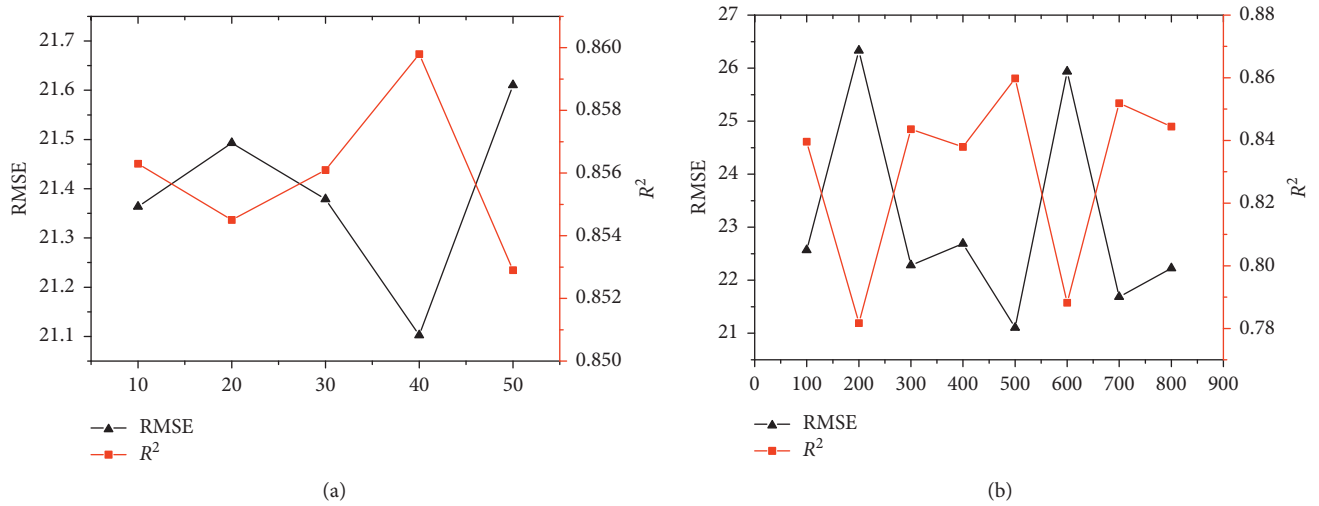


Figure 8: Optimization process of hyperparameters for LSTM: (a) number of hidden layer nodes; (b) number of batch sizes.

Table 2: The RMSE of different methods.

Test	CNN	LSTM	CNN-LSTM
#1	76.4694	26.3256	22.8220
#2	68.1422	26.0771	22.2389
#3	105.1664	25.6918	22.8374
#4	66.4166	25.4898	26.5900
#5	80.0437	26.0154	23.0910
#6	66.0526	25.8325	23.9463
#7	135.0926	25.7496	27.4446
#8	95.9767	25.7942	21.9647
#9	91.5383	25.8024	23.1579
#10	53.0630	27.0787	22.9959
Average	83.7962	26.2065	23.7089

Table 3: The R^2 of different methods.

Test	CNN	LSTM	CNN-LSTM
#1	0.4935	0.7931	0.8222
#2	0.4718	0.7912	0.7923
#3	0.3329	0.7923	0.8060
#4	0.4891	0.7946	0.8367
#5	0.1761	0.7980	0.8358
#6	0.3739	0.7900	0.8014
#7	0.4748	0.7999	0.8130
#8	0.1901	0.7964	0.8448
#9	0.1094	0.7734	0.8452
#10	0.1133	0.5345	0.8328
Average	0.3224	0.7663	0.8237

LSTM was more accurate in NO_x emission prediction with average RMSE of 23.7089 and R^2 of 0.8237. The combination of CNN and LSTM integrates the advantages of CNN and LSTM which are capable of extracting the features among different data sequences and the features between different time steps. The ability of CNN-LSTM is suitable for the characteristics of datasets from refining and chemical enterprises. By describing the local feature relationship under multidimensional and long-term conditions, CNN-LSTM matches the observations better than the other methods.

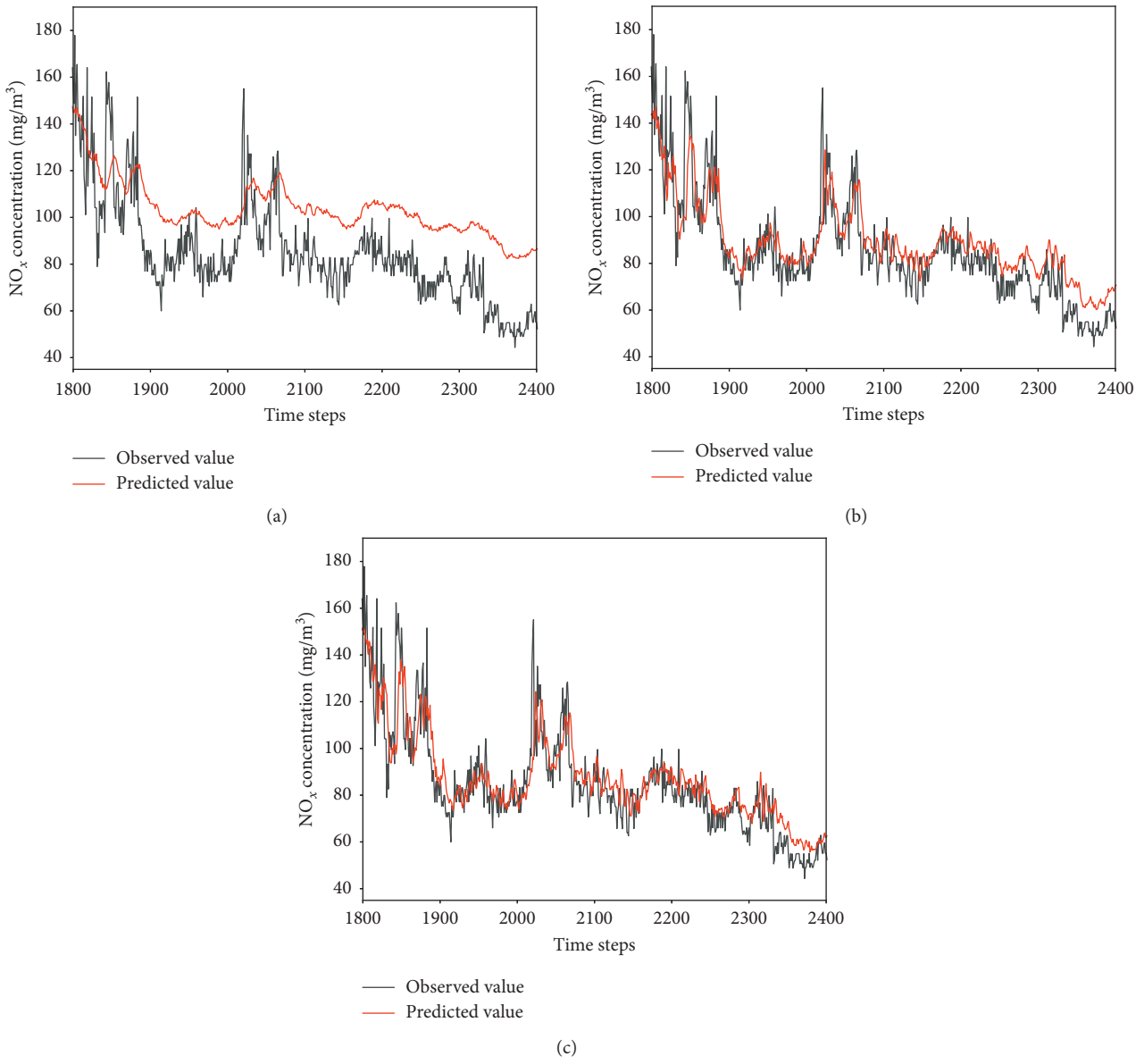


Figure 9: Comparisons of the observed and predicted NO_x concentrations of (a) CNN; (b) LSTM; (c) CNN-LSTM.

6. Conclusions

In this paper, a novel CNN-LSTM scheme combining CNN and LSTM was proposed for the prediction of NO_x concentration observed during FCC process. Dropout were introduced to accelerate network training and address the overfitting issue. In our study, a series of hyperparameters (learning rate, regularization parameter, the number of neurons in each hidden layer, small batch data size, convolution kernel size, neuron activation function, pool layer size, and dropout rate) and conditions (raw materials, process control parameters of reactor, regeneration process control parameters, and catalyst species) were selected and optimized. Experiments were conducted to evaluate the proposed scheme with traditional methods (CNN and LSTM) being baseline models. The hyperparameters of all the methods were optimized to obtain the best results. RMSE and R^2 were used to evaluate the performance of different methods. Due to the capability of extracting features among different data sequences and different time steps, better efficiency and accuracy were obtained by CNN-LSTM than baseline models. This study provides a potential direction of deep learning methods by integrating different architectures for individual advantages. The CNN-LSTM scheme proposed in this paper would be a beneficial contribution to the accurate and stable prediction of irregular trends for NO_x emission from refining industry and provided more reliable information for NO_x risk assessment and management. Future work will focus on attention and transformer mechanism to obtain better results and explore the application of the proposed scheme on other datasets.

Data Availability

All data and program files included in this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by CNPC Basic Research Fund Projects "Research on Prediction and Early Warning System of Air Pollution Emission in FCC and New Control Model" (no. 2017D-5008) and Science Foundation of China University of Petroleum, Beijing (nos. 2462018YJRC007 and 2462020YXZZ025).

References

[1] L. A. Lacijan, M. W. Schnaith, P. J. Van Opdorp et al., "FCC refinery solutions for the European market," in *Petroleum Technology Quarterly*, Palo Alto, CA, USA, 2002.

[2] I. V. Babich, K. Seshan, and L. Lefferts, "Nature of nitrogen specie in coke and their role in NO_x formation during FCC catalyst regeneration," *Applied Catalysis B: Environmental*, vol. 59, no. 3-4, pp. 205–211, 2005.

[3] P. Li, L. K. Zeng, H. Wang, and X. S. Cheng, "Technological feasibility for controlling NO_x emission of ceramic furnace," *Chinese Academy of Sciences*, vol. 22, no. 3, pp. 35–42, 2015.

[4] J. L. Lin, X. B. Hua, Y. Q. Wu, and C. Yang, "Generation and control of nitrogen oxides in flue gas of catalytic plant regeneration," *Environmental Protection in Petrochemical Industry*.vol. 28, no. 1, pp. 34–38, 2005.

[5] R. Ehrlich and M. C. Henry, "Chronic toxicity of nitrogen dioxide," *Archives of Environmental Health: An International Journal*, vol. 17, no. 6, pp. 860–865, 1968.

[6] J. Buick, R. Lowry, and T. R. Magee, "Pulmonary aspects of nitrogen dioxide toxicity," *Resources, Conservation and Recycling*, vol. 27, no. 1-2, pp. 89–97, 1999.

[7] K. Sertić-Bionda, Z. Gomzi, and M. Mužic, "Modeling of gas oil catalytic cracking in a fixed bed reactor using a five-lump kinetic model," *Chemical Engineering Communications*, vol. 197, no. 3, pp. 275–288, 2009.

[8] C. Jia, S. Rohani, and A. Jutan, "FCC unit modeling, identification and model predictive control, a simulation study," *Chemical Engineering and Processing: Process Intensification*, vol. 42, no. 4, pp. 311–325, 2003.

[9] P. Kiss and F. Szeifert, "Coordinated control of a fluid catalytic cracking unit," *Chemical Engineering & Technology*, vol. 21, no. 6, pp. 515–521, 1998.

[10] M. V. Cristea, Ş. P. Agachi, and V. Marinoiu, "Simulation and model predictive control of a UOP fluid catalytic cracking unit," *Chemical Engineering and Processing: Process Intensification*, vol. 42, no. 2, pp. 67–91, 2003.

[11] X. Dupain, E. D. Gamas, R. Madon, C. P. Kelkar, M. Makkee, and J. A. Moulijn, "Aromatic gas oil cracking under realistic FCC conditions in a microriser reactor," *Fuel*, vol. 82, no. 13, pp. 1559–1569, 2003.

[12] I. S. Nam and J. R. Kittrell, "Use of catalyst coke content in deactivation modeling," *Industrial & Engineering Chemistry Process Design and Development*, vol. 23, no. 2, pp. 237–242, 1984.

[13] C. I. C. Pinheiro, J. L. Fernandes, L. Domingues et al., "Fluid catalytic cracking (FCC) process modeling, simulation, and control," *Industrial & Engineering Chemistry Research*, vol. 51, no. 1, pp. 1–29, 2012.

[14] D. V. Naik, V. Karthik, V. Kumar, B. Prasad, and M. O. Garg, "Kinetic modeling for catalytic cracking of pyrolysis oils with VGO in a FCC unit," *Chemical Engineering Science*, vol. 170, no. 12, pp. 790–798, 2017.

[15] J. Long, M. S. Mao, and G. Y. Zhao, "Model optimization for an industrial fluid catalytic cracking riser-regenerator unit by differential evolution algorithm," *Petroleum Science and Technology*, vol. 33, no. 13-14, pp. 1380–1387, 2015.

[16] S. Sun, H. Yan, and F. Meng, "Optimization of a fluid catalytic cracking kinetic model by improved particle swarm optimization," *Chemical Engineering & Technology*, vol. 43, no. 2, pp. 289–297, 2020.

[17] J. Michalopoulos, S. Papadokonstadakis, G. Arampatzis, and A. Lygeros, "Modelling of an industrial fluid catalytic cracking unit using neural networks," *Chemical Engineering Research and Design*, vol. 79, no. 2, pp. 137–142, 2001.

[18] G. M. Bollas, S. Papadokonstadakis, J. Michalopoulos et al., "Using hybrid neural networks in scaling up an FCC model from a pilot plant to an industrial unit," *Chemical Engineering and Processing: Process Intensification*, vol. 42, no. 8-9, pp. 697–713, 2003.

[19] Z. X. Zhang, *Utility Boiler Combustion Optimization Based on Intelligent Optimization Algorithm*, North China Electric Power University, Beijing, China, 2015.

[20] Y. P. Gu, W. J. Zhao, and Z. S. Wu, "Combustion optimization for utility boiler based on least squares support vector machine," *Proceedings of the CSEE*, vol. 30, no. 17, pp. 91–97, 2010.

[21] B. Shao, M. Li, Y. Zhao, and G. Bian, "Nickel price forecast based on the LSTM neural network optimized by the improved PSO algorithm," *Mathematical Problems in Engineering*, vol. 2019, Article ID 1032480, 15 pages, 2019.

[22] F. Yang, C. Dai, J. Tang, J. Xuan, and J. Cao, "A hybrid deep learning and mechanistic kinetics model for the prediction of fluid catalytic cracking performance," *Chemical Engineering Research and Design*, vol. 155, no. 1, pp. 202–210, 2020.

[23] G. Miskell, W. Pattinson, L. Weissert, and D. Williams, "Forecasting short-term peak concentrations from a network of air quality instruments measuring PM2.5 using boosted gradient machine models," *Journal of Environmental Management*, vol. 242, pp. 56–64, 2019.

[24] K. Zhao, T. He, S. Wu et al., "Application research of image recognition technology based on CNN in image location of environmental monitoring UAV," *EURASIP Journal on Image and Video Processing*, vol. 2018, no. 1, pp. 1–11, 2018.

[25] W.-Y. Chen, Y.-F. Liao, and S.-H. Chen, "Speech recognition with hierarchical recurrent neural networks," *Pattern Recognition*, vol. 28, no. 6, pp. 795–805, 1995.

[26] L. Gao, P.-Y. Chen, and S. Yu, "Demonstration of convolution kernel operation on resistive cross-point array," *IEEE Electron Device Letters*, vol. 37, no. 7, pp. 870–873, 2016.

[27] F. Y. Zhou, L. P. Jin, and J. Dong, "Review of convolutional neural network," *Chinese Journal of Computers*, vol. 40, no. 2, 2017.

[28] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate Shift," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, vol. 37, pp. 448–456, Lille, France, July 2015.

[29] G. E. Hinton, N. Srivastava, A. Krizhvsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," 2012, https://arxiv.org/abs/1207.0580.

[30] F. Dan Foresee and M. T. Hagan, "Gauss-Newton approximation to Bayesian learning," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 3, pp. 1930–1935, IEEE, Houston, TX, USA, June 1997.

[31] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machined," in *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 807–814, Haifa, Israel, June 2010.

[32] O. Russakovsky, J. Deng, H. Su et al., "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS)*, pp. 1097–1105, Lake Tahoe, NV, USA, December 2012.

[34] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[35] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber, "LSTM: a search space odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, 2017.

[36] P. Zhang, "Mean-mean absolute deviation portfolio model and optimization," *Statistics and Decision-Making*, vol. 1, pp. 14-15, 2009.