Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 6842835, 13 pages
https://doi.org/10.1155/2021/6842835

Research Article

Data-Driven Quality Prediction of Batch Processes Based on Minimal-Redundancy-Maximal-Relevance Integrated Convolutional Neural Network

Yufeng Dong,[1,2] Yingping Zhuang,[1] and Xuefeng Yan [1,2]

[1]*State Key Laboratory of Bioreactor Engineering, East China University of Science and Technology, Shanghai 200237, China*
[2]*Key Laboratory of Smart Manufacturing in Energy Chemical Process, Ministry of Education, East China University of Science and Technology, Shanghai 200237, China*

Correspondence should be addressed to Xuefeng Yan; xfyan@ecust.edu.cn

Received 1 June 2021; Accepted 11 November 2021; Published 27 November 2021

Academic Editor: Paolo Spagnolo

For batch processes that are extensively applied in modern industry and characterized by nonlinearity and dynamics, quality prediction is significant to obtain high-quality products and maintain production safety. However, some quality variables and key performance indicators are difficult to measure online. In addition, the mechanism-based model for batch processes is usually tough to acquire due to the strong nonlinearity and dynamics, which makes quality prediction a challenge. With the accumulation of historical process data, data-driven methods for quality prediction gain increasing attention, among which convolutional neural network (CNN) is quite successful for its automatic feature extraction of nonlinear features from raw data. Considering that most CNN-based methods mainly take the variety of extracted features into account and ignore the redundancy between them, this paper introduces the minimal-redundancy-maximal-relevance algorithm to select features obtained by original CNN and further improves it with a feature selection layer to form the proposed method referred as mRMR-CNN. Then, a quality prediction model is established based on mRMR-CNN and the effectiveness of it is verified on the penicillin fermentation process, where the proposed method shows remarkable performance.

1. Introduction

In modern industry, batch processes are extensively applied to the production of high-value products in many areas such as pharmaceutical, biotechnology, and polymer and semiconductor manufacturing industries [1–4]. Due to complex process mechanism, process uncertainties, and special production requirements, batch processes are characterized by strong nonlinearity and dynamics. For such nonlinear dynamic processes, it is quite prominent to ensure high-quality products and safely running production process, which makes it necessary to monitor quality variables or key performance indicators of the process. However, constrained by hard sensors or economic efficiency, such variables or indicators are usually difficult to measure [5]. To address this issue, a generic scheme is used to establish a prediction model to estimate the value of quality variables using easy-to-measure variables in the process, which facilitates the existence of numerous quality prediction methods.

The methods for quality prediction are mainly categorized into two types, mechanism-based methods and data-driven methods [6]. Due to nonlinearity and complexity of batch processes, it is essentially tough to establish accurate mathematical models, which greatly inhibit the development of mechanism-based approaches to batch processes [7]. With the accumulation of industrial historical data, which are considered to cover adequate information of batch processes, data-driven methods grow into the mainstream of academia, estimating the value of target quality variables by mainly seeking information from historical data.

With respect to data-driven methods, they are mainly referred to statistic methods and machine learning methods [8], which include partial least squares (PLS) [9], supported vector regression (SVR) [10, 11], back-propagation (BP) neural network [12, 13], and autoencoder (AE) [14–16]. However, such methods with shallow structures may not fit the complex and large-scale industrial data well. Then, with deep learning gaining increasing attention, there exist methods with deep structures, such as stacked autoencoder (SAE) [17, 18], deep belief network (DBN) [14, 19], and convolutional neural network (CNN) [20]. Among all these methods, CNN is well known for its excellent automatic feature extraction capability and is successfully applied to many fields, such as image recognition, computer vision, and natural language processing [21]. Owing to its outstanding performance and extensive applications, CNN is selected as the basis of the proposed method in this study.

When applying CNN to quality prediction, it is common to form a 2D matrix input by segmenting time-series data composed of different variables alongside the sampling time. Basically, there exist both time and variable axes in the input data fed into the CNN model, in sense of which the CNN model can not only consider the relationship between local adjacent variables at the same sampling points but also consider the relationship between local adjacent sampling points of the same variables. It indicates that CNN can extract local features contained in historical data from both the spatial (or variable) and temporal perspectives. Therefore, CNN can extract various features from industrial historical data, for which it is common to be of extensive applications in quality monitoring. Wei et al. [22] developed a soft sensor model based on CNN and extreme learning machine (ELM) to measure fill level inside ball mill, fully exploiting vibration frequency spectrum. Sun et al. [23] proposed a virtual sensor model using CNN to predict dynamic responses in structural systems and obtain relatively high accuracy. Zhu et al. [24] applied CNN to predicting next time step measurements and utilized moving window to cover time-dependent correlation information, which achieved remarkable performance on industrial pyrolysis reactor. Shevchik et al. [25] used spectral CNN to classify data collected by acoustic emission sensors, thus achieving quality monitoring in additive manufacturing. Olivier et al. [26] employed CNN to characterize the feed size of run-of-mine ore and trained the constructed model using transfer learning to reduce required data. Wang et al. [27] took process dynamics into consideration and integrated finite impulse response with CNN, which can effectively improve the prediction accuracy. Jiang et al. [28] presented a semi-supervised soft sensor to balance the labeled and unlabeled data by constructing 2D data and used CNN to extract spatial information contained in the 2D data. Yuan et al. [29] comprehensively considered local correlations between variables to form a multichannel CNN for soft sensing, which significantly improved the estimation accuracy. In addition, Yuan et al. [30] also proposed a dynamic CNN to learn hierarchical dynamic nonlinear feature, based on which they developed a soft sensor model exploring both the spatial and temporal correlations from industrial data.

However, all these methods try to extract abundant discriminative features, while the redundancy between extracted features is neglected, which may degrade the performance of CNN-based models. It is quite a common contradiction. On the one hand, it is desired that numerous features can be extracted by CNN from historical data, which indicates the necessity of various representations to form an effective predictor. On the other hand, it cannot be guaranteed that all the obtained features are independent of each other. Furthermore, if certain dependent features cannot provide enough extra useful information to the prediction of target variables, they may even add noises [31]. Considering that eliminating irrelevant and redundant features can benefit the improvement of model learning performance [32], it is essential to enable CNN the capability of selecting features to behave better in quality prediction.

Although some works [33–35] suggest that L1 regularization can be added to CNN to make the structure sparse, which can implicitly perform feature selection, it is not straightforward and may not guarantee the required prediction precision. To take feature redundancy into account and select features under guidance, an algorithm named minimal-redundancy-maximal-relevance (mRMR) [36] is introduced to the original CNN model; that is, a CNN model is pretrained to extract plenty of features and then mRMR algorithm is applied to extracted features to form a feature subset that can best fit the model so that the original CNN can be enriched with a feature selection layer, after which the modified model will be retrained to promote the performance.

The main contributions of this paper are summarized into the following three points:

(1) A CNN model, LeNet-5, is employed as the baseline to automatically extract adequate nonlinear discriminative features from given data.

(2) mRMR algorithm is applied to select extracted features and a feature selection layer is then integrated with the original CNN model to form the proposed method, noted as mRMR-CNN, so that feature redundancy is taken into consideration to improve the performance of original CNN model.

(3) A quality prediction model based on mRMR-CNN is established and the effectiveness of proposed mRMR-CNN method is validated on the penicillin fermentation process, where the proposed method distinguishes itself with considerable increase in quality prediction precision.

The remainders of this paper are arranged as follows. Section 2 basically introduces primary conceptions of CNN and mRMR. Then, Section 3 discusses the proposed method in detail, while Section 4 conducts an experiment to validate the proposed method. The final section draws a conclusion of this paper.

2. Related Works

The basic knowledge of CNN and mRMR is briefly introduced in this section.

2.1. CNN. CNN is a successful deep learning technique proposed by Lecun et al. [20], which simulates the mechanism of cat's visual cortex [37]. Moreover, it is well known for its excellent automatic feature extraction ability and extensively applied to image classification, computer vision, and natural language processing fields. Recent years have seen the arising of multitudinous CNN models, such as AlexNet [38], VGGNet [39], GoogLeNet [40], ResNet [41], and DenseNet [42], all of them exhibiting elite performance.

However, this subsection is about to discuss the time-tested fundamental CNN structure, namely, LeNet-5, on which the approach proposed in this paper is based. The basic structure of it is depicted in Figure 1.

As illustrated in Figure 1, LeNet-5 is composed of two parts, a feature extractor and a classifier (for classification)/regressor (for regression). The former part consists of an input layer, alternating adjacent convolutional layer, and subsampling layer (or pooling layer), in charge of feature extraction from input data. The latter part contains a couple of fully connected layers, performing classification/regression tasks based on features obtained by the extractor [43].

It is obvious that CNN is essentially a type of multilayer feedforward artificial neural networks (ANNs). However, it is unique for its convolutional layer, pooling layer, and usually 2D input data when compared to conventional ANNs [44].

The convolutional layer, which typically contains several feature mappings, makes CNN special with sharing weights and local receptive field [20]. More specifically, in the convolutional layer, each feature mapping is connected to the previous layer through a different square convolution kernel; thus, each unit of the same feature mapping shares the weights contained in the related convolution kernel. Due to such sparse connections and considering that the size of a convolution kernel is much smaller than that of the input, each unit of the feature mapping in the current layer merely corresponds to a small zone of original input image data, which interprets local receptive field. It is notable that all these connections rely on convolution operation, which is to express below. Assume that there is an input feature mapping with size of $w_i \times h_i$ and a convolution kernel with size of $f \times f$. s defines the sliding stride of convolution while there is no zero padding. The dimension $w_o \times h_o$ of the output feature mapping obtained by convolution operation can be calculated using the following equations (1) and (2):

$$w_o = \frac{(w_i - f)}{s + 1}, \tag{1}$$

$$h_o = \frac{(h_i - f)}{s + 1}. \tag{2}$$

Then, the basic convolution operation is visually illustrated in Figure 2 and precise value $o_{i,j}$ at coordinate (i, j) in output feature mapping can be mathematically expressed with the following equation:

$$o_{i,j} = \sigma\left(\sum_{k=1}^{f}\sum_{l=1}^{f} w_{k,l} x_{(i\times s+k-1),(j\times s+l-1)} + b\right), \quad 1 \le i \le w_o, 1 \le j \le h_o, \tag{3}$$

where $w_{k,l}$ represents the weight value at coordinate (k, l) of the convolution kernel, while b refers to the bias value. $x_{(i\times s+k-1),(j\times s+l-1)}$ is the unit response at coordinate $(i \times s + k - 1,\ j \times s + l - 1)$ of input feature mapping. σ denotes the activation function, as which rectified linear unit (ReLU) function is usually designated in CNN.

As for the subsampling layer, which is as well referred as the pooling layer, it generally follows the convolutional layer, performing pooling operation to reduce spatial resolution and data dimension of feature mappings derived from the preceding layer [20]. There exist mainly two types of pooling operations, average pooling and max pooling, the latter of which is the most used one. When max pooling functions, each feature mapping of the previous convolutional layer is implicitly divided into a couple of 2×2 square areas, known as pooling fields, with no overlapping and no gap. Then, the maximum value of responses in each area is calculated to form the corresponding unit of the current pooling layer. Thereby, the number of feature mappings in the pooling layer is identical to that in the prior convolutional layer. The max pooling is intuitively illustrated in Figure 3.

2.2. mRMR. mRMR is an effective feature selection model proposed for classification tasks by Peng et al. [36], which is based on mutual information between variables.

Assume two random variables X and Y, to which $p(x)$, $p(y)$, and $p(x, y)$ are the probabilistic density functions related. Then, the mutual information $I(X; Y)$ between the two variables can be defined by the following formula:

$$I(X; Y) = \sum_{x\in X}\sum_{y\in Y} p(x, y)\log\frac{p(x, y)}{p(x)p(y)}. \tag{4}$$

When implementing mRMR, the purpose is to find a feature subset S with m features, which is the solution of the following optimization problem:

$$\max\ D - R, \tag{5}$$

in which the specific expression of D and R is given in the following formulas:

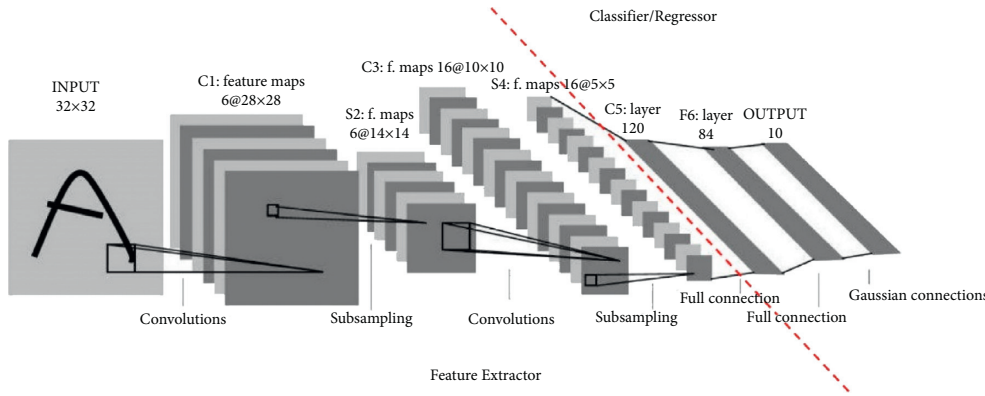

Figure 1: The structure of LeNet-5 [20].

$$D = \frac{1}{|S|} \sum_{x_i \in S} I(x_i; c), \tag{6}$$

$$R = \frac{1}{|S|^2} \sum_{x_i, x_j \in S} I(x_i; x_j), \tag{7}$$

where $|S|$ indicates the feature number of set S. D is defined to quantify the average relevance between each feature variable x_i and related class c, while R denotes the quantitative redundancy between feature variables when the feature subset is given. Therefore, to optimize the problem defined by (5) is essentially maximizing the relevance between variables and related class while minimizing the redundancy among variables, under the guidance of which unnecessary features are screened out and a feature subset that can fit the target task well is reserved.

Although mRMR is originally developed for classification tasks, it works for regression tasks as well. The difference lies in calculating the mutual information between dependent variable x and independent variable y, instead of variables x and related class c, to quantify corresponding relevance D. This makes the relevance D slightly different in formula as the following expression:

$$D = \frac{1}{|S|} \sum_{x_i \in S} I(x_i; y). \tag{8}$$

3. Proposed Method

Due to nonlinearity and complexity of batch processes, it is necessary to extract abundant features from the historical data to cover the most discriminative ones that can fit the desired estimation well, in sense of which the obtained features are usually more than it genuinely requests. However, traditional CNN lacks the ability to directly perform feature selection. Then, how to make CNN able to decrease feature redundancy is a crucial problem to be handled. Although several works report that L1 regularization can make sparse the structure of CNN and implicitly attain feature reduction, it is not intuitive and may not guarantee prediction precision. Hence, this paper proposes an improved CNN model combined with minimal-redundancy-maximal-relevance, referred as mRMR-CNN to straightly select the extracted features so that a feature selection layer is added to original CNN, of whom the performance can be improved.

Specifically, a CNN model is initially constructed and pretrained slightly overfitting the given training samples, which empirically implies that the built CNN model achieves relatively acceptable performance and last convolutional (or pooling) layer of it covers numerous discriminative features. Since these features are automatically extracted and not necessarily independent of each other, then mRMR serves here performing feature selection to reduce redundant information. mRMR takes CNN-extracted features and expected model outputs as algorithm input. When feature preservation proportion is given, mRMR derives from the extracted features a feature subset that can fit given data well and have the least redundancy. Feature preservation proportion is used here to indicate the number of features to be reserved, which should be experimentally determined. After mRMR functions, a feature selection layer is added into CNN, closely following the last convolutional (or pooling) layer, to achieve feature reduction. The feature selection layer mainly contains a constant weight matrix of size $N_e \times N_s$ filled with 0 and 1, where N_e is the number of CNN-extracted features and N_s is the number of mRMR-selected features. Each column of the weight matrix functions to pick out one feature; thus, each column contains only one element 1, while the rest elements are all zeros. Finally, the modified CNN, mRMR-CNN, is retrained to attain better performance. Figure 4 illustrates the main idea of proposed method and divides it into pretraining and retraining parts.

In addition, the detailed procedures of proposed mRMR-CNN are described as follows:

(1) Step 1: preprocess raw data samples, scale them to [0, 1], and convert them into 2D matrix data.

(2) Step 2: construct a classical CNN framework inheriting the structure of LeNet-5.

(3) Step 3: keep revising the parameters and structure of the constructed model and pretrain the model to a relatively acceptable performance so that most of the discriminative features can be obtained.

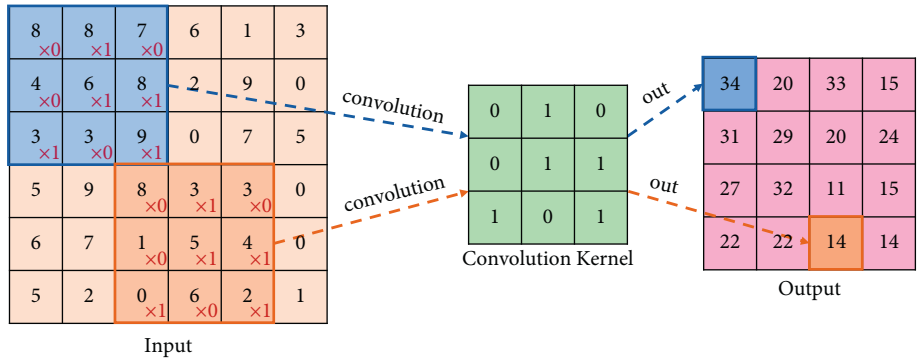

Figure 2: Convolution with convolution kernel of size 3×3, stride of 1, and no zero padding.

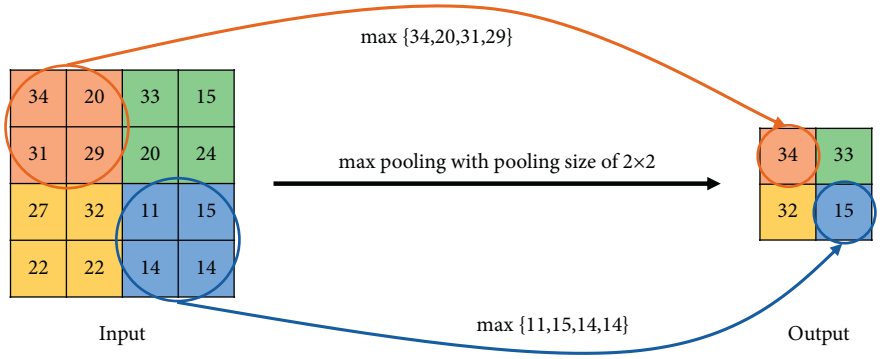

Figure 3: Visual illustration of max pooling.

(4) Step 4: recover features generated in the last convolutional layer (or pooling layer) of CNN model constructed in step 1, and perform mRMR algorithm to determine the features to be maintained when given the preservation proportion or percentage.

(5) Step 5: based on the results of step 4, add a feature selection layer right between the last convolutional layer and the first fully connected layer. Then, retrain and fine-tune the modified CNN model, mRMR-CNN.

(6) Step 6: use the grid search approach to determine the best feature subset and eventually fix the structure of mRMR-CNN owning best performance, which is a loop to repeat steps 3–5 until all the given feature preservation proportions are examined.

To further state the proposed method, the final structure of mRMR-CNN is briefly depicted in Figure 5, and a more detailed architecture of it will be displayed in Table 1 of Section 4.2. In the proposed mRMR-CNN, the main difference from original CNN is an additional feature selection layer succeeding the feature extractor, which is determined by mRMR algorithm and CNN-extracted features.

Based on the aforementioned presentation of proposed method, a quality prediction model is established for batch processes and the scheme of it is depicted in Figure 6, which consists of two phases, training phase and testing phase. The training phase mainly adheres to the procedures of proposed method to determine the structure of mRMR-CNN, while the testing phase is simply an application of fixed mRMR-CNN.

4. Case Study

The effectiveness of mRMR-CNN on quality prediction modeling is verified on the penicillin fermentation process with simulation software PenSim v2.0 [45]. Two widely used indices, RMSE and the coefficient of determination R^2, are introduced to evaluate the experiment results. The mathematical expression of two indices is defined as follows:

$$\text{RMSE} = \sqrt{\frac{1}{N_T}\sum_{i=1}^{N_T}\left(y_i - \hat{y}_i\right)^2},$$
$$R^2 = 1 - \frac{\sum_{i=1}^{N_T}\left(y_i - \hat{y}_i\right)^2}{\sum_{i=1}^{N_T}\left(y_i - \overline{y}\right)^2}, \tag{9}$$

where N_T stands for the sample number. y_i represents the real value, while $\hat{y}_i$ denotes the estimation value, and $\overline{y}$ is noted as the mean of all the real values.

To implement the experiment, all codes are written in *Python* 3.7 under deep learning framework TensorFlow 2.1.0 except that mRMR-concerned codes are written in MATLAB. All programs are carried out in Windows 10 64 bit enterprise edition with Intel (*R*) Xeon (*R*) Sliver 4110 CPU @ 2.10 GHz, 32.0 GB RAM, and NVIDIA Quadro P620 2 GB GPU.

4.1. Penicillin Fermentation Process and Experiment Configuration. The penicillin fermentation process is a benchmark widely used for validating the performance of

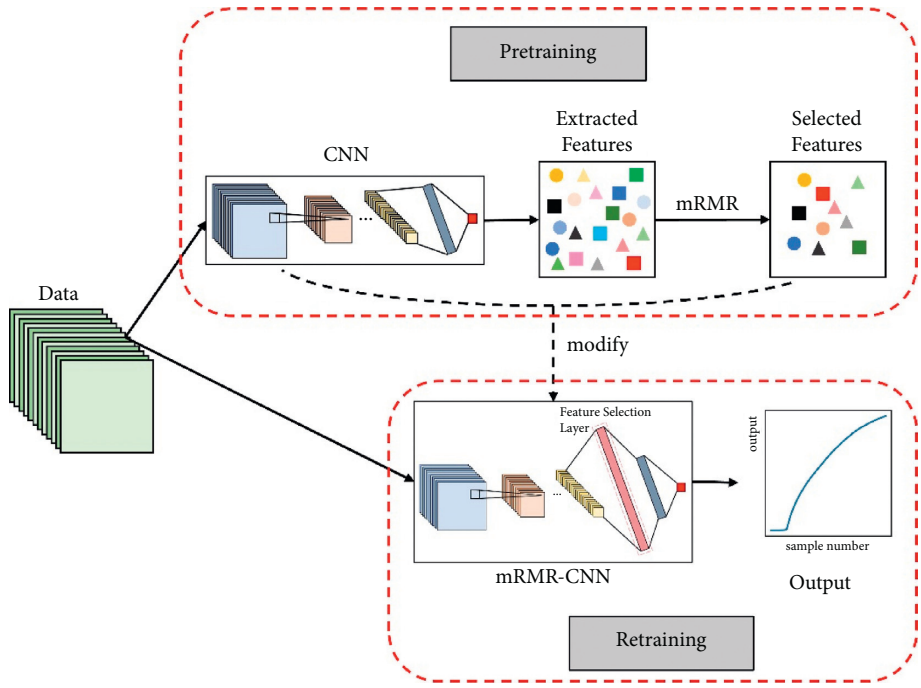

Figure 4: Illustration of mRMR-CNN.

batch process quality prediction modeling. The simulation software of it can be downloaded from http://www.chee.iit.edu/~control/software.html. Figure 7 displays the flowchart of penicillin fermentation process.

Based on certain works [45–47], difficult-to-measure penicillin concentration is the key quality variable of penicillin fermentation process and the rest sixteen variables are regarded as auxiliary variables, which involve aeration rate (L/h), agitator power (W), substrate feed rate (L/h), substrate feed temperature (K), substrate concentration (g/L), dissolved oxygen concentration (g/L), biomass concentration (g/L), culture volume (L), carbon dioxide concentration (mmol/L), pH, fermenter temperature (K), generated heat (cal), acid flow rate (L/h), base flow rate (L/h), cold water flow rate (L/h), and hot water flow rate (L/h). 25 groups of data, totally 10000 samples, are obtained with parameter settings suggested by related works [46–48]. The obtained data are divided into training data set of 8000 samples and testing data set of 2000 samples according to production batches. Among training samples, 7200 of them are randomly selected for training the constructed model and the rest for validating and fine-tuning the model.

4.2. Structure Determination of mRMR-CNN. This subsection attempts to determine the structure of mRMR-CNN with the best performance and the well-known dropout technique is adopted in the fully connected layer to ensure the prediction capability, which is as well applied to other neural network-based methods used in this paper, if not specified. 960 features are extracted by pretrained CNN, and the prominent grid search approach is executed using feature preservation proportion grid [0.20, 0.25, 0.30, 0.35, 0.40, 0.50, 0.55, 0.60, 0.65, 0.70]. Under each proportion, the corresponding model is trained 10 times with training epochs of 300, training batch size of 128, and ReLU as default activation function. Adam's algorithm is adopted to achieve error back propagation and learn the trainable parameters. Specifically, learnable parameters, including weights in convolution kernel and fully connected layer, can be iteratively updated using the following equations:

$$
\begin{aligned}
g_i &= \frac{\partial L}{\partial w}\Big|_{w_{i-1}}, \\
m_i &= \beta_1 m_{i-1} + (1-\beta_1) g_i, \\
v_i &= \beta_2 v_{i-1} + (1-\beta_2) g_i^2, \\
\widehat{m}_i &= \frac{m_i}{\left(1-\beta_1^i\right)}, \\
\widehat{v}_i &= \frac{v_i}{\left(1-\beta_2^i\right)}, \\
w_i &= w_{i-1} - \frac{\alpha \cdot \widehat{m}_i}{\left(\widehat{v}_i + \varepsilon\right)},
\end{aligned}
\tag{10}
$$

where i is the iteration index, w is the weight to update, L is the loss function, m is the first momentum variable, and v is the second raw momentum variable. Meanwhile, α is the learning rate, ε is an extremely small positive number in case of zero division, and β_1 and β_2 are the exponential decay rates for the momentum variables. Here, hyper-parameters use the default settings, $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\varepsilon = 10^{-7}$.

Then, the average validating prediction values on test data set are adopted as the final prediction values, on which the calculation of RMSE and R^2 is based. If not specified, all

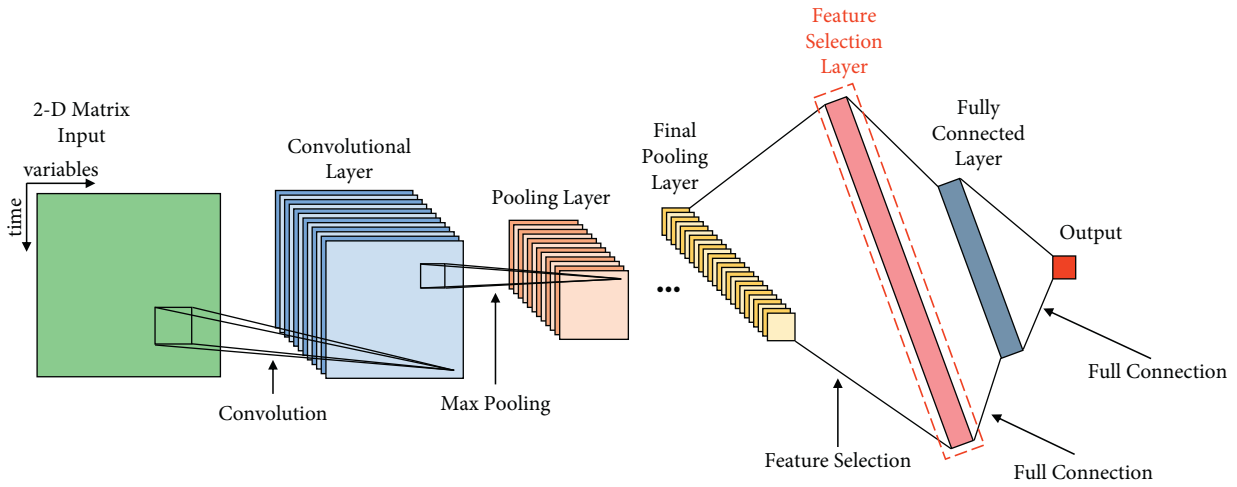

Figure 5: The structure of mRMR-CNN.

Table 1: Specific structure of fixed mRMR-CNN.

Type	Kernel size/stride	Output size	Parameters	Trainable
Convolution	$3 \times 3/1$	$50 \times 16 \times 10$	100	True
Max pooling	$2 \times 2/2$	$25 \times 8 \times 10$	—	—
Convolution	$3 \times 3/1$	$25 \times 8 \times 20$	1820	True
Max pooling	$2 \times 2/2$	$12 \times 4 \times 20$	—	—
Feature selection	—	336	336	False
Dropout (50%)	—	336	—	—
Full-connection	—	40	13480	True
Dropout (50%)	—	40	—	—
Output	—	1	41	True

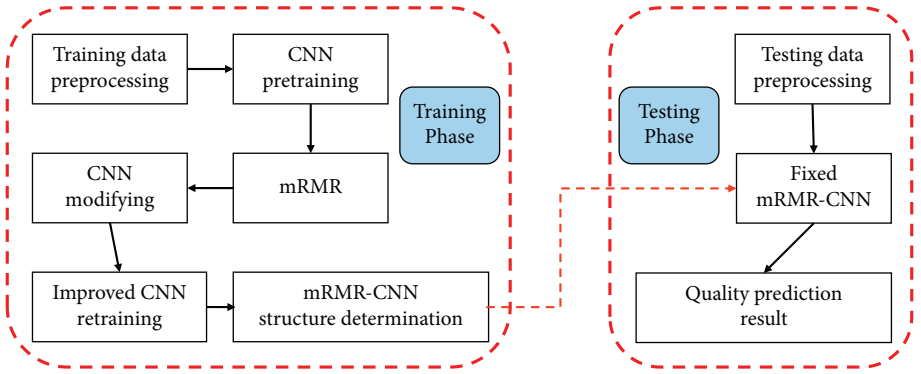

Figure 6: Scheme of quality prediction modeling based on mRMR-CNN.

neural network-based models employed in this study take the same strategy.

Detailed RMSE and R^2 values under varied feature preservation proportions are given in Table 2, where the best results are highlighted in bold font. The results indicate that validating RMSE value reaches the lowest and coefficient of determination R^2 value attains highest when feature preservation proportion is set as 0.35, which means the mRMR-CNN achieves its best performance under current given conditions. Then, the specific structure of finally fixed mRMR-CNN is offered in Table 1. It is noticeable that feature selection essentially decreases nodes connected to those in the first fully connected layer. To put it differently, the number of trainable parameters in the first fully connected layer will increase to 38,440 if the feature selection layer is dropped. Therefore, there is actually a 64.9% reduction in trainable parameters in the first fully connected layer when compared with CNN.

To visually present the validating results, Figures 8 and 9 are drawn to display part of the prediction results and prediction errors under different feature preservation proportions, respectively. In Figure 8, the closer the scattered points (painted in blue) distribute to the reference line (the red solid

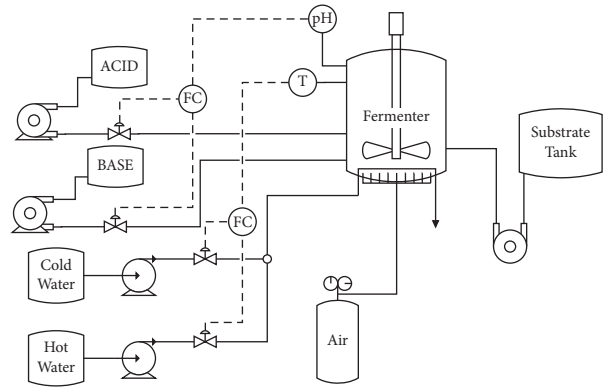

Figure 7: Flowchart of penicillin fermentation process.

Table 2: Validating RMSE and R^2 under varied feature preservation proportions.

proportion	0.20	0.25	0.30	0.35	0.40	0.45	0.50	0.55	0.60	0.65	0.70	1.00
RMSE	0.0423	0.0375	0.0336	**0.0249**	0.0327	0.0359	0.0391	0.0397	0.0423	0.0435	0.0468	0.0506
R^2	0.9692	0.9758	0.9805	**0.9893**	0.9816	0.9777	0.9736	0.9728	0.9691	0.9673	0.9621	0.9558

Best results are highlighted in bold font.

one), the better prediction results. In such sense, prediction results are fairly acceptable when preserving 35.0% CNN-extracted features, where scattered points are the closest to the reference line. Figure 9 is presented to show the fluctuation of prediction results, where prediction errors vary in smaller scale under feature preservation proportion 0.35 when compared to other proportions. It implies that prediction results are relatively stable and precise under proportion 0.35.

4.3. Comparison between Different Methods. To further validate the performance of proposed method, it is compared with SVR, AE, BP, SAE, original CNN, and CNN-L1, which applies L1 regularization to the first fully connected layer. SVR is a traditional machine learning method, and the rest are neural network-based methods.

In this paper, the basic parameter settings for SVR are sensitivity ϵ of 0.08, penalty factor C of 1000, and using radial basis function (RBF) with kernel coefficient γ of 0.02 as kernel function. BP is a three-layer neural network (excluding input layer) with (336, 40, 1) nodes in each layer. AE shares the structure of 16-13-16 and SAE stacks two AEs, following the structure of 16-13-11-13-16. The features learned by AE and SAE are then fed into a three-layer neural network with 384-64-1 structure to regress the target variable. It is notable that both ReLU and sigmoid functions are employed in AE and SAE. The specific structure of mRMR-CNN here is the same as listed in Table 1. Baseline CNN is similar to the mRMR-CNN, merely removing the feature selection layer. CNN-L1 mainly distinguishes itself from CNN with L1 regularizer adopted in the first fully connected layer, and regularization coefficient is set as 0.0001. All neural network-based methods here are trained using training epochs of 300 and Adam to optimize trainable parameters. Other relevant hyper-parameters here are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\varepsilon = 10^{-7}$, where α is the learning rate.

RMSE and R^2 of training and testing with different approaches are listed in Tables 3 and 4, respectively. The proposed mRMR-CNN outperforms other methods on testing data set with the lowest RMSE value of 0.0249 and highest R^2 value of 0.9893, while SVR behaves worst on both training and testing data sets. It is worth noting that CNN reaches almost the same performance as mRMR-CNN on the training data set but performs 3.4% inferior to mRMR-CNN on the testing data set in terms of R^2, and mRMR-CNN behaves 50.8% superior to CNN on testing samples in terms of RMSE. The performance of CNN-L1 is acceptable on both training and testing samples but slightly worse than that of mRMR-CNN. To justify the performance results, Table 5 provides the prediction time per sample consumed by CNN, CNN-L1, and mRMR-CNN when testing. It can be seen that mRMR-CNN makes predictions slightly faster than the other two approaches.

Additionally, prediction results of different methods are illustrated in Figures 10 and 11. In Figure 10, a shorter average distance of scattered points to the reference line indicates better prediction results. It can be seen that scattered points under mRMR-CNN distribute densely close to the reference line, while those of other approaches distribute either sparser or even far away from the reference line. In Figure 11, tracking results given by SVR, BP, AE, and CNN-L1 exhibit several sharp leaps when real output grows from zero. Meanwhile, SAE seems to make predictions with noises, resulting in sustaining small fluctuation. In contrast, the tracking curve obtained by CNN and mRMR-CNN is smoother and more stable. Especially, mRMR-CNN obtains predictions closer to the real values.

To further investigate the prediction results, the prediction errors and their corresponding distribution under different methods are shown in Figures 12 and 13 ,

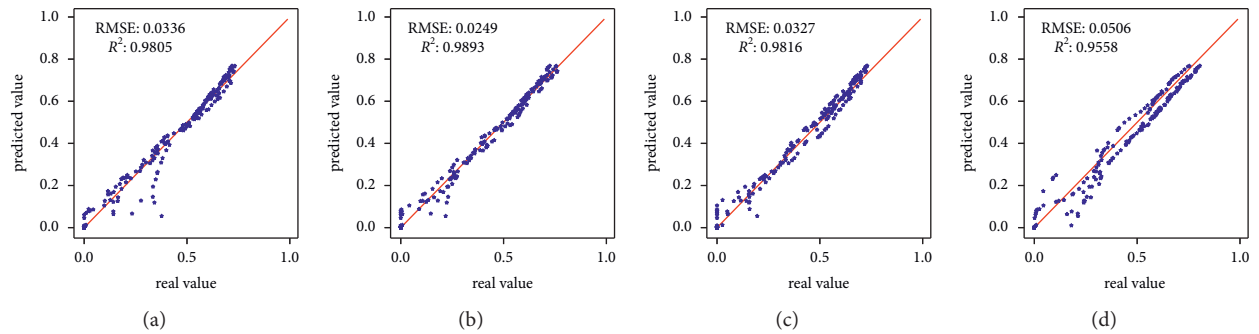

Figure 8: Part of the prediction results under different feature preservation proportions. (a) 30.0% features reserved, (b) 35.0% features reserved, (c) 40.0% features reserved, and (d) all features reserved.

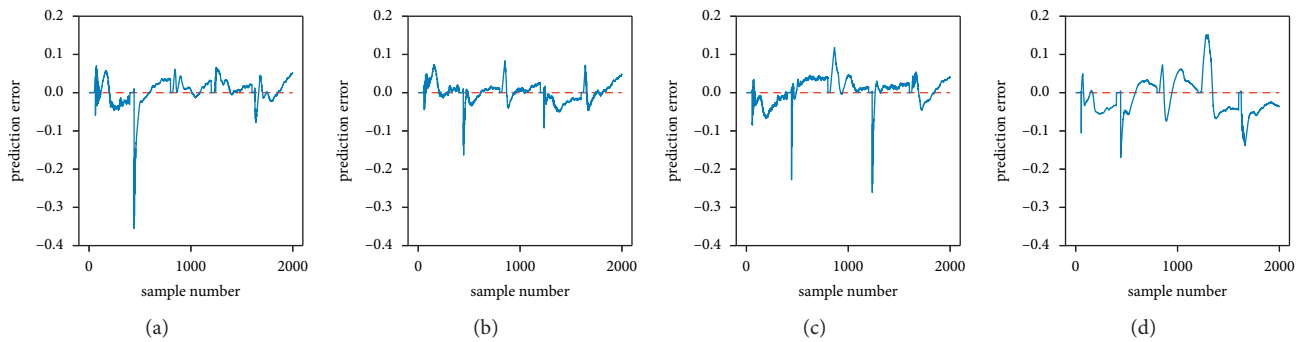

Figure 9: Part of the prediction errors under different feature preservation proportions. (a) 30.0% features reserved, (b) 35.0% features reserved, (c) 40.0% features reserved, and (d) all features reserved.

Table 3: Training RMSE and R^2 of different methods.

Methods	SVR	BP	AE	SAE	CNN	CNN-L1	mRMR-CNN
RMSE	0.0571	0.0297	0.0270	0.0397	0.0246	0.0293	**0.0232**
R^2	0.9560	0.9881	0.9901	0.9788	0.9918	0.9884	**0.9927**

Bold indicates the lowest RMSE value and highest R^2 value.

Table 4: Testing RMSE and R^2 of different methods.

Methods	SVR	BP	AE	SAE	CNN	CNN-L1	mRMR-CNN
RMSE	0.0735	0.0495	0.0407	0.0614	0.0506	0.0359	**0.0249**
R^2	0.9071	0.9579	0.9716	0.9351	0.9558	0.9778	**0.9893**

Bold indicates the lowest RMSE value and highest R^2 value.

Table 5: Testing prediction time consumed by three methods: CNN, CNN-L1, and mRMR-CNN.

Methods	CNN	CNN-L1	mRMR-CNN
Seconds per sample	6.0057×10^{-5}	6.0600×10^{-5}	$\mathbf{6.0005 \times 10^{-5}}$

Bold font indicates the best result, which indicates the shortest testing prediction time.

respectively. From Figure 12, prediction errors of mRMR-CNN have the smallest fluctuation scale, ranging from −0.15 to 0.1, while those of most other methods are even down to −0.3. From Figure 13, mRMR-CNN obtains the narrowest box plot. In addition, the mean of prediction errors (the blue dash line in the tiny box) nearly equals the median (the red solid line), close to zero, in box plot for mRMR-CNN. It indicates prediction errors of mRMR-CNN distribute roughly around zero.

From the above results, mRMR-CNN outperforms some traditional methods (such as SVR, BP, and AE) and even certain deep learning methods (such as SAE, CNN, and

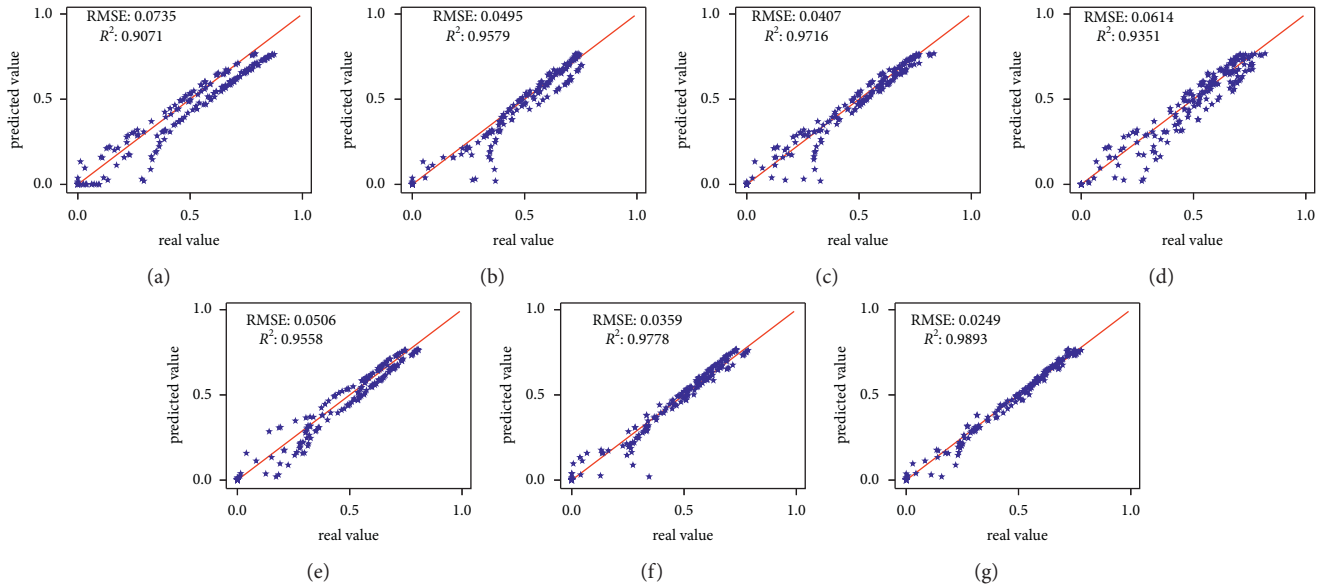

Figure 10: Prediction results under varied methods. (a) SVR, (b) BP, (c) AE, (d) SAE, (e) CNN, (f) CNN-L1, and (g) mRMR-CNN.

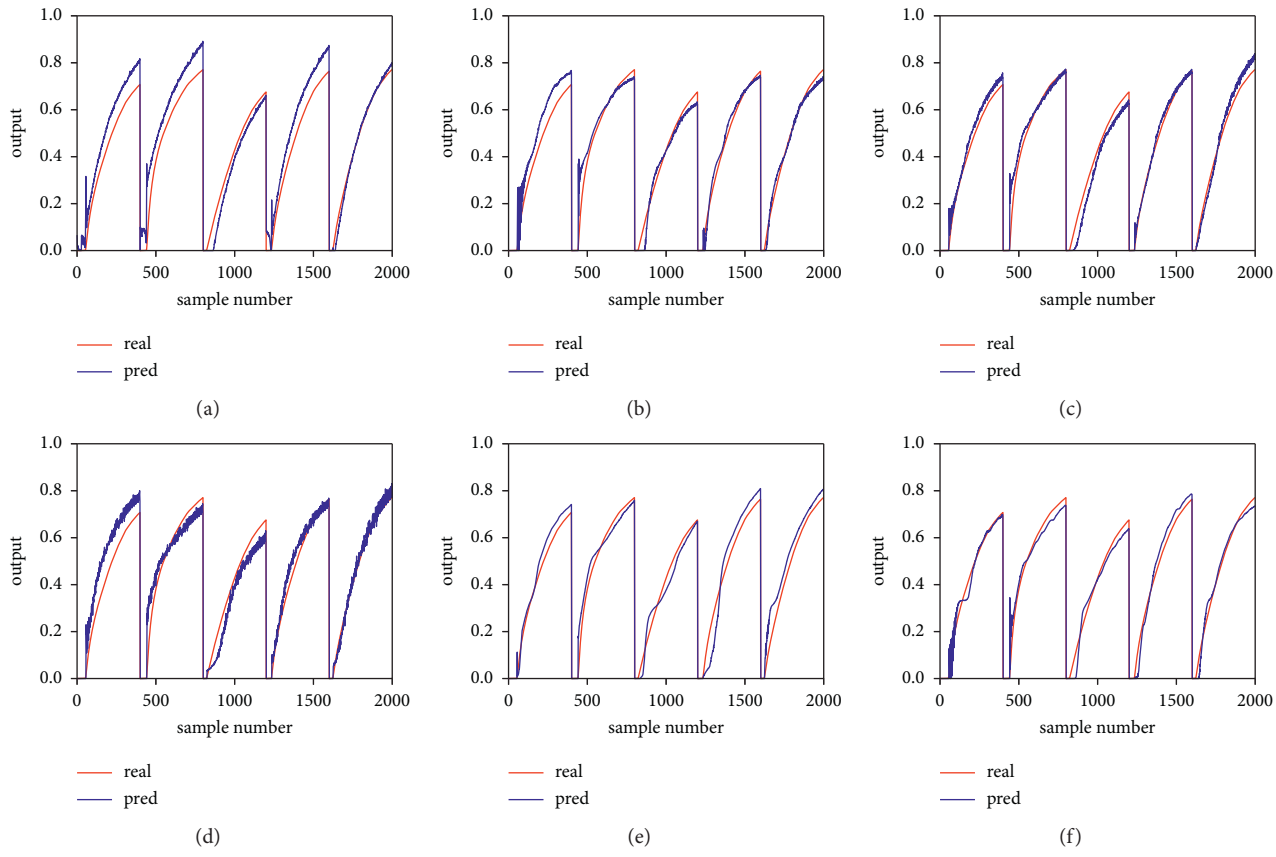

Figure 11: Continued.

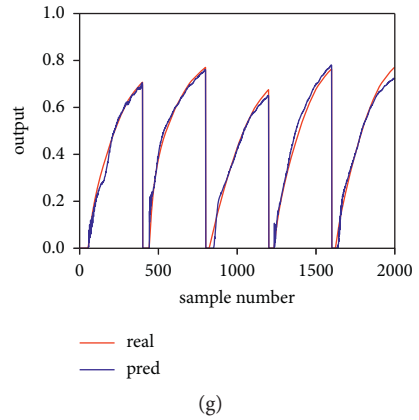

Figure 11: Prediction tracking on real value under different methods. (a) SVR, (b) BP, (c) AE, (d) SAE, (e) CNN, (f) CNN-L1, and (g) mRMR-CNN.

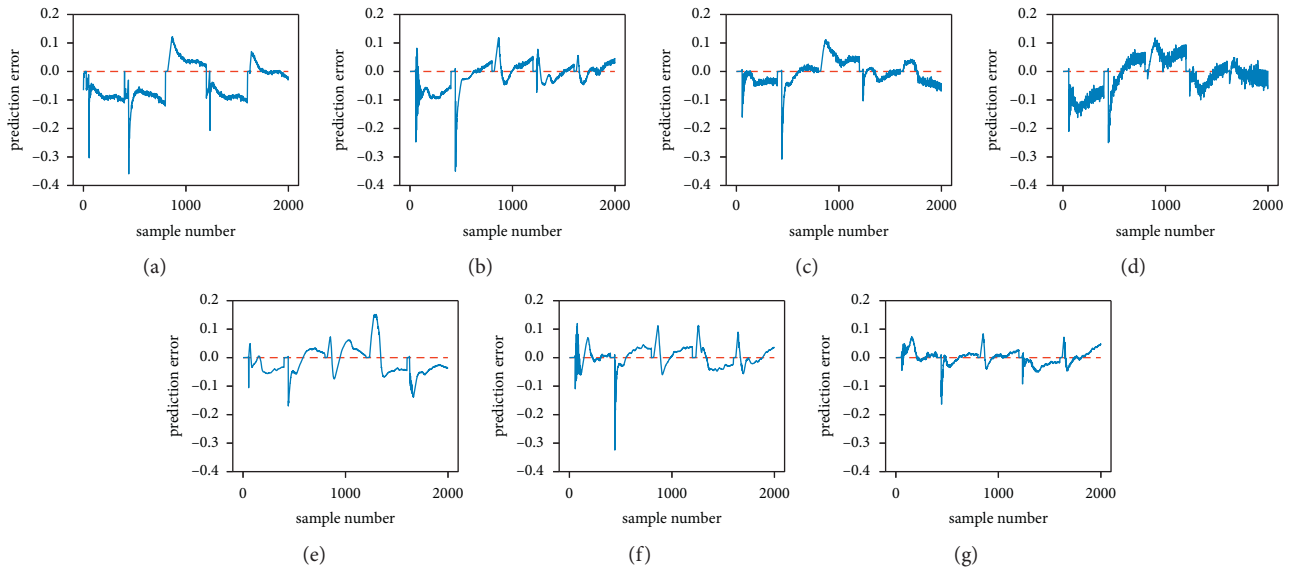

Figure 12: Prediction errors of different methods. (a) SVR. (b) BP. (c) AE. (d) SAE. (e) CNN. (f) CNN-L1. (g) mRMR-CNN.

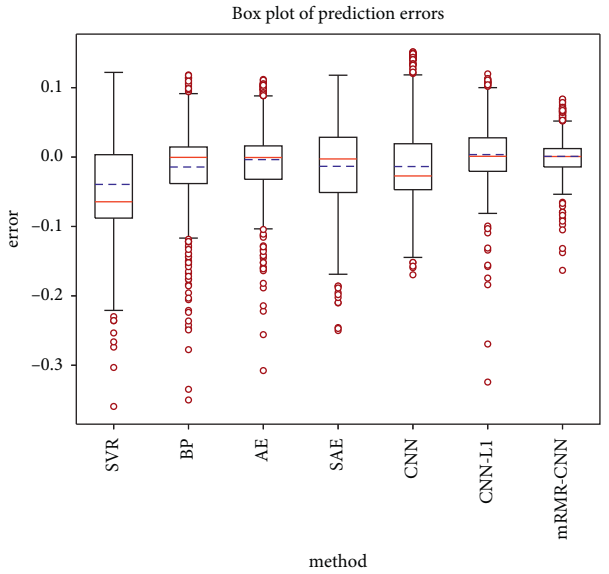

Figure 13: Box plot of prediction errors under different methods.

CNN-L1), which could be owed to the excellent feature extraction ability of CNN and the explicit feature selection in the training stage of mRMR-CNN. Other approaches except CNN-L1 fail either to capture valid features or to reduce feature redundancy, resulting in unsatisfactory performance. Although CNN-L1 obtains acceptable performance, it suffers subtle instability in prediction, probably due to implicit feature selection. Based on the above analyses, it can be concluded that mRMR-CNN, which explicitly facilities CNN with the ability of feature selection, can effectively promote quality prediction precision. It implies as well the significance of feature selection in CNN-based model, in consideration of feature redundancy.

5. Conclusion

In this study, a novel mRMR-CNN approach is proposed to model quality prediction for batch processes, of which the key idea is using mRMR to remove redundant features obtained by original CNN so that a feature selection layer can be added to the original CNN to enhance the quality prediction precision. Then, the performance of proposed method is verified on the penicillin fermentation process, where the results indicate that the proposed method achieves a significant improvement when compared to the original CNN. Furthermore, in terms of penicillin fermentation process, the performance of proposed method is superior to that of certain methods such as CNN-L1, SAE, and SVR. Additionally, mRMR-CNN can also greatly decrease trainable parameters when compared to the original CNN, although it is not the concern of this study.

However, as mentioned before, the training stage of proposed method is divided into two phases, pretraining phase and retraining phase. Therefore, the training time that the proposed method consumes is obviously more than the traditional CNN and CNN-L1 methods. Then, the future focus may lie in how to dynamically integrate mRMR into the training stage of CNN to enhance model performance and shorten training time.

Data Availability

The training and testing data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Acknowledgments

This work was mainly funded by the National Key Research and Development Program of China (2021YFC2101100) and National Natural Science Foundation of China (21878081) and was partially supported by the Open Funding Project of the State Key Laboratory of Bioreactor Engineering. The authors appreciate the support of the above organizations.

References

[1] C. Ündey, S. Ertunç, and A. Çınar, "Online batch/fed-batch process performance monitoring, quality prediction, and variable-contribution analysis for diagnosis," *Industrial & Engineering Chemistry Research*, vol. 42, no. 20, pp. 4645–4658, 2003.

[2] S. Yin, S. X. Ding, A. H. Abandan Sari, and H. Hao, "Data-driven monitoring for stochastic systems and its application on batch process," *International Journal of Systems Science*, vol. 44, no. 7, pp. 1366–1376, 2013.

[3] Q. Jiang, X. Yan, H. Yi, and F. Gao, "Data-driven batch-end quality modeling and monitoring based on optimized sparse partial least squares," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 5, pp. 4098–4107, 2020.

[4] F. Shen, J. Zheng, L. Ye, and D. Gu, "Quality-relevant monitoring of batch processes based on stochastic programming with multiple output modes," *Processes*, vol. 8, no. 2, 2020.

[5] X. Yan, J. Wang, and Q. Jiang, "Deep relevant representation learning for soft sensing," *Information Sciences*, vol. 514, pp. 263–274, 2020.

[6] C. Shang, F. Yang, D. Huang, and W. Lyu, "Data-driven soft sensor development based on deep learning technique," *Journal of Process Control*, vol. 24, no. 3, pp. 223–233, 2014.

[7] Q. Jiang, S. Yan, X. Yan, H. Yi, and F. Gao, "Data-driven two-dimensional deep correlated representation learning for nonlinear batch process monitoring," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 4, pp. 2839–2848, 2020.

[8] W. Yan, R. Xu, K. Wang, T. Di, and Z. Jiang, "Soft sensor modeling method based on semisupervised deep learning and its application to wastewater treatment plant," *Industrial & Engineering Chemistry Research*, vol. 59, no. 10, pp. 4589–4601, 2020.

[9] L. Luo, S. Bao, J. Mao, and D. Tang, "Quality prediction and quality-relevant monitoring with multilinear PLS for batch processes," *Chemometrics and Intelligent Laboratory Systems*, vol. 150, pp. 9–22, 2016.

[10] X. Gao, W. Pu, and C. Sun, "Modeling for penicillin fermentation process based on support vector machine," *Journal of System Simulation*, vol. 07, pp. 2052–2055, 2006.

[11] J. Feng, Y. Tang, and T. Peng, "Prediction model of penicillin fed-batch fermentation based on KTA-LSSVM," *Chemical Industry and Engineering Progress*, vol. 33, no. 9, pp. 2438–2443, 2014.

[12] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[13] L. Li and P. Feng, "Modeling of penicillin fermentation process based on the combination of nondominated sorting genetic algorithm II and error back propagation network," *Computers and Applied Chemistry*, vol. 25, no. 11, pp. 1334–1336, 2008.

[14] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[15] S. Yan and X. Yan, "Using labeled autoencoder to supervise neural network combined with k-nearest neighbor for visual industrial process monitoring," *Industrial & Engineering Chemistry Research*, vol. 58, no. 23, pp. 9952–9958, 2019.

[16] X. Gao, Z. Xu, Z. Li, and P. Wang, "Batch process monitoring using multiway Laplacian autoencoders," *Canadian Journal of Chemical Engineering*, vol. 98, no. 6, pp. 1269–1279, 2020.

[17] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," *Advances in Neural Information Processing Systems*, vol. 19, pp. 153–160, 2006.

[18] Z. Li, L. Tian, Q. Jiang, and X. Yan, "Distributed-ensemble stacked autoencoder model for non-linear process monitoring," *Information Sciences*, vol. 542, pp. 302–316, 2021.

[19] P. Lian, H. Liu, X. Wang, and R. Guo, "Soft sensor based on DBN-IPSO-SVR approach for rotor thermal deformation prediction of rotary air-preheater," *Measurement*, vol. 165, 2020.

[20] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[21] A. Dhillon and G. K. Verma, "Convolutional neural network: a review of models, methodologies and applications to object detection," *Progress in Artificial Intelligence*, vol. 9, no. 2, pp. 85–112, 2020.

[22] J. Wei, L. Guo, X. Xu, and G. Yan, "Soft Sensor Modeling of Mill Level Based on Convolutional Neural Network," in *Proceedings of the 27th Chinese Control and Decision Conference (2015 CCDC)*, pp. 4738–4743, Qingdao, China, May 2015.

[23] S. B. Sun, Y. Y. He, S. D. Zhou, and Z.-J. Yue, "A data-driven response virtual sensor technique with partial vibration measurements using convolutional neural network," *Sensors*, vol. 17, no. 12, 2017.

[24] W. Zhu, Y. Ma, Y. Zhou, M. Benton, and J. Romagnoli, "Deep learning based soft sensor and its application on a pyrolysis reactor for compositions predictions of gas phase components," in *Computer Aided Chemical Engineering*, M. R. Eden, M. G. Ierapetritou, and G. P. Towler, Eds., vol. 44pp. 2245–2250, 2018.

[25] S. A. Shevchik, C. Kenel, C. Leinenbach, and K. Wasmer, "Acoustic emission for in situ quality monitoring in additive manufacturing using spectral convolutional neural networks," *Additive Manufacturing*, vol. 21, pp. 598–604, 2018.

[26] L. E. Olivier, M. G. Maritz, and I. K. Craig, "Deep convolutional neural network for mill feed size characterization," *IFAC-PapersOnLine*, vol. 52, no. 14, pp. 105–110, 2019.

[27] K. Wang, C. Shang, L. Liu, Y. Jiang, D. Huang, and F. Yang, "Dynamic soft sensor development based on convolutional neural networks," *Industrial & Engineering Chemistry Research*, vol. 58, no. 26, pp. 11521–11531, 2019.

[28] X. Jiang, L. Yao, G. Huang et al., "A spatial-information-based semi-supervised soft sensor for f-CaO content prediction in cement industry," in *Proceedings of the 2020 IEEE 9th Data Driven Control and Learning Systems Conference*, pp. 898–905, Liuzhou, China, November 2020.

[29] X. Yuan, S. Qi, Y. A. W. Shardt, Y. Wanga, C. Yanga, and W. Gui, "Soft sensor model for dynamic processes based on multichannel convolutional neural network," *Chemometrics and Intelligent Laboratory Systems*, vol. 203, 2020.

[30] X. Yuan, S. Qi, Y. Wang, and H. Xia, "A dynamic CNN for nonlinear dynamic feature learning in soft sensor modeling of industrial process data," *Control Engineering Practice*, vol. 104, Article ID 104614, 2020.

[31] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16–28, 2014.

[32] R. Sheikhpour, M. A. Sarram, S. Gharaghani, and M. A. Z. Chahooki, "A survey on semi-supervised feature selection methods," *Pattern Recognition*, vol. 64, pp. 141–158, 2017.

[33] P. Kulkarni, J. Zepeda, F. Jurie, P. Pérez, and L. Chevallier, "Learning the structure of deep architectures using L1 regularization," in *Proceedings of the 2015 British Machine Vision Conference*, pp. 23.1–23.11, Swansea, Wales, September 2015.

[34] J. M. Alvarez and M. Salzmann, "Learning the number of neurons in deep networks," *Advances in Neural Information Processing Systems*, vol. 29, pp. 2270–2278, 2016.

[35] R. Ma, J. Miao, L. Niu, and P. Zhang, "Transformed ℓ1 regularization for learning sparse deep neural networks," *Neural Networks*, vol. 119, pp. 286–298, 2019.

[36] P. Hanchuan Peng, L. Fuhui Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[37] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of Physiology*, vol. 160, no. 1, pp. 106–154, 1962.

[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, 2012.

[39] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *Computer Vision and Pattern Recognition*, ArXiv Preprint arXiv:1409.1556, 2014.

[40] C. Szegedy, W. Liu, Y. Jia et al., "Going deeper with convolutions," in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, Boston, MA, USA, June 2015.

[41] K. He, X. Zhang, and S. Ren, "Deep residual learning for image recognition," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, Las Vegas, NV, USA, June 2016.

[42] G. Huang, Z. Liu, and L. Van Der Maaten, "Densely Connected Convolutional Networks," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4700–4708, Honolulu, HI, USA, July 2017.

[43] K. B. Lee, S. Cheon, and C. O. Kim, "A convolutional neural network for fault classification and diagnosis in semiconductor manufacturing processes," *IEEE Transactions on Semiconductor Manufacturing*, vol. 30, no. 2, pp. 135–142, 2017.

[44] L. Wen, X. Li, L. Gao, and Y. Zhang, "A new convolutional neural network-based data-driven fault diagnosis method," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 7, pp. 5990–5998, 2018.

[45] G. Birol, C. Ündey, and A. Çinar, "A modular simulation package for fed-batch fermentation: penicillin production," *Computers & Chemical Engineering*, vol. 26, no. 11, pp. 1553–1565, 2002.

[46] Y. Liu and H. Wang, "Pensim simulator and its application in penicillin fermentation process," *Journal of System Simulation*, vol. 18, no. 12, 2006.

[47] L. Ye and J. Cheng, "Simulator of penicillin fermentation process in Matlab/Simulink environment," *Journal of System Simulation*, vol. 27, no. 3, pp. 515–520, 2015.

[48] W. Fu, *Research on Prediction of Penicillin Concentration by Using Multi-Model Approach Based on RBF Neural Network*, Northeastern University, Shenyang, China, 2010.