Deep Convolutional Neural Network with Deconvolution and a Deep Autoencoder for Fault Detection and Diagnosis

Yasuhiro Kanno and Hiromasa Kaneko*

Cite This: ACS Omega 2022, 7, 2458−2466

ABSTRACT: In chemical plants and other industrial facilities, the rapid and accurate detection of the root causes of process faults is essential for the prevention of unknown accidents. This study focused on deep learning while considering the different phenomena that can occur in industrial facilities. A deep convolutional neural network with deconvolution and a deep autoencoder (DDD) is proposed. DDD assesses the process dynamics and the nonlinearity between process variables. During the operation of DDD, fault detection is carried out using the reconstruction error between the data reconstructed through the model and the input data. After a process fault is detected, the magnitude of the contribution of each process variable to the detected process fault is calculated by applying gradient-weighted class activation mapping to the established network. The effectiveness of DDD in fault detection and diagnosis was verified through experiments on the Tennessee Eastman process dataset, demonstrating that it can achieve improved performance compared to the conventional fault detection and diagnosis.

1. INTRODUCTION

In chemical plants, accidents and mechanical failures result in significant economic losses. In such environments, process variables, such as temperature, flow rates, and pressure, are measured constantly to monitor and control chemical plant operations while ensuring the safety of the equipment as well as that of the human resources. 1 It is also possible to monitor chemical plant processes by applying techniques based on statistical process control (SPC), which rely on the variable data of measured processes. The number of process variables required to account for all the physical and chemical phenomena increases as the processes become increasingly complex. Therefore, monitoring multiple process variables collectively using multivariate SPC (MSPC) is considered a highly efficient approach. As a result, various MSPC-based methodologies 7 have been proposed. Examples of statistical methods that rely on MSPC include principal component analysis (PCA),8 independent component analysis,9 partial least squares analysis,10 artificial neural networks,11 and support vector machines.12 In addition, deep neural networks have been receiving increased research attention because the networks can be used to express complex relationships between various process variables.

Using neural networks with multiple layers, it is possible to realize the deeper learning of features contained in the data in a stepwise manner. Deep learning-based approaches, such as the deep autoencoder (DAE)8 and convolutional neural networks (CNNs),9 are used to detect and classify faults in chemical processes10 and motor bearing.11 Such approaches are also employed in other various fields and applications such as the diagnosis of malfunctions, including bearing failures12 and turbine failures.13 However, the interpretation of constructed deep neural networks can be difficult. This is despite their necessity in identifying the root causes of process faults in chemical processes following the detection of such faults.

Therefore, in the field of artificial intelligence, several methods for clarifying the bases estimated using deep neural networks have been proposed. For example, CNNs are currently considered effective approaches in the field of image processing. CNNs incorporate a visualization method known as gradient-weighted class activation mapping (Grad-CAM)14 whose main function involves clarifying the basis for a judgment. Grad-CAM is a method for calculating the part of a neural network that contributes the most to a specific output class using the gradient of the convolutional layer and
the probability score. This method has been used in the classification of pig models and MRI brain images to establish a basis for the classification of Alzheimer’s disease. However, although conventional methods can be applied in supervised learning applications, such as those for image classification, the methods cannot be applied in unsupervised learning applications, such as those for detecting process faults in chemical plants. Therefore, it is crucial to detect process faults and diagnose such faults.

The aim of this study was to develop a method for detecting process faults using a deep neural network. In this study, a method that combines a CNN with a DAE is proposed to consider the nonlinearity between process variables and process dynamics in process variables. Because CNNs can consider the pixel intensity as well as the spatial relationship between pixels, it is possible to extract the temporal characteristics of each process variable. Subsequent dimensional reduction was performed using a DAE, after which the latent variables considering the nonlinearity of the variables were extracted. Process faults were detected and diagnosed using the data transformed through the model (similar to PCA-based MSPC) and the reconstruction error of the input data (similar to Q in PCA-based MSPC). This methodology is referred to as a deep convolutional neural network with deconvolution and a deep autoencoder (DDD). By applying Grad-CAM in the constructed neural network, it is possible to detect and diagnose process faults in latent variables by considering the pixel intensity as well as the spatial relationship between pixels, it is possible to extract the temporal characteristics of each process variable. Subsequent dimensional reduction was performed using a DAE, after which the latent variables considering the nonlinearity of the variables were extracted. Process faults were detected and diagnosed using the data transformed through the model (similar to PCA-based MSPC) and the reconstruction error of the input data (similar to Q in PCA-based MSPC). This methodology is referred to as a deep convolutional neural network with deconvolution and a deep autoencoder (DDD). By applying Grad-CAM in the constructed neural network, it is possible to detect and diagnose process faults in latent variables by considering the nonlinearity of the variables.

2. METHODS

The proposed method, DDD, combines a DAE with a CNN, and fault diagnosis using DDD is based on the incorporation of Grad-CAM. First, DAE, CNN, and Grad-CAM are explained, and then DDD and fault diagnosis with DDD are discussed.

2.1. Deep Autoencoder. A basic autoencoder (AE) is a neural network comprising three layers, namely, an input layer, a single hidden layer, and an output layer. A network model comprising multiple hidden layers is referred to as a DAE. Figure 1 shows a schematic diagram of an AE. Given the input data \( x \in \mathbb{R}^{n \times u} \), where \( n \) represents the number of samples and \( u \) represents the number of process variables \( X \), the input samples \( x \in \mathbb{R}^{u} \), \( i = 1,2, ..., n \) are encoded into neurons \( h_i \in \mathbb{R}^{r} \), where \( v \) represents the number of neurons in the hidden layer, using the following formula:

\[
\mathbf{h}_i = f(\mathbf{W}_i \mathbf{x}_i + \mathbf{b}_i)
\]

where \( \mathbf{W}_i \in \mathbb{R}^{r \times u} \) and \( \mathbf{b}_i \in \mathbb{R}^{r} \) represent the weight and bias in the encoding process, respectively. Subsequently, the input sample is decoded from the neurons \( h_i \in \mathbb{R}^{r} \) into the reconstructed sample \( \hat{x} \in \mathbb{R}^{u} \) using the following formula:

\[
\hat{x}_i = f(\mathbf{W}_i \mathbf{h}_i + \mathbf{b}_i)
\]

where \( \mathbf{W}_i \in \mathbb{R}^{u \times r} \) and \( \mathbf{b}_i \in \mathbb{R}^{u} \) represent the weight and bias in the decoding process, respectively. Therefore, the reconstructed data \( \hat{x} \in \mathbb{R}^{u} \) of \( X \) are obtained from the AE. \( f \) represents an activation function that extracts the input features, and the sigmoid, tanh, and rectified linear unit (ReLU) functions are used as general activation functions. The AE is trained so that the reconstruction error between \( X \) and \( \hat{X} \) diminishes, and \( \theta = \{ \mathbf{W}_i, \mathbf{b}_i, \mathbf{W}_j, \mathbf{b}_j \} \) is updated using the backpropagation method, as shown in the following equation:

\[
\min f_{AE} = \frac{1}{2n} \sum_{i=1}^{n} \| \hat{x}_i - x_i \|^2
\]

2.2. Convolutional Neural Network. A CNN is a neural network that comprises an input layer, a convolutional layer, a pooling layer, a deconvolution layer, and an output layer. The convolutional layer applies a defined number of filters to obtain a feature map of the input image, and the pooling layer reduces the number of input features. The convolutional and pooling layers are alternately repeated several times to extract the final number of features. The image reconstructed through the deconvolution layer is then the output.

2.2.1. Convolutional Layer. The output of the convolutional layer comprises feature maps in which each unit is connected to a local patch of the input feature map via a weighted filter. All the units in the output feature map share the same filter, and within a layer, different feature maps use different filters. The convolutional layer can be used to facilitate the detection or recognition of patterns present in the process data. Assuming that there are \( M \times 1 \) input feature maps \( x \) in the \( l \)th layer and \( N \) filters, the output feature map \( x^{l+1} \) at the \( j \)th position in the \( l + 1 \)th layer is calculated as follows:

\[
x_j^{l+1} = f \left( \sum_{i=1}^{M} x_i^l k_{ij} + b_j \right), j = 1, ..., N
\]

where \( k_{ij} \) represents the kernel of the \( j \)th filter connected to the \( i \)th input feature map \( x_i^l \), \( x^{l+1} \) represents the \( i \)th input’s feature map, \( x^{l+1} \) represents the \( j \)th input’s feature map, \( f \) represents the activation function, and the asterisk symbol (*) represents the convolution operation. Common activation functions for neural networks include the sigmoid, tanh, and ReLU functions. Assuming a kernel size of \( s \times s \), the number of all parameters in the convolutional layer is calculated as follows:

\[
P = N \times (s \times s \times M + 1)
\]

The output feature map obtained from the convolutional layer is transferred to the pooling layer.

2.2.2. Pooling Layer. The pooling layer follows the convolutional layer and downsamples the input feature map. The purpose of the pooling layer is to compress information and transform the input data into a more manageable form. The use of pooling layers has two advantages. First, because the relative positions of the features forming the local pattern
may differ slightly, detecting the features with similar local positions offers enhanced reliability. Second, the dimensionality of feature representation can be reduced without setting parameters, thereby significantly reducing the computation time and the parameters of the entire network.

Primarily, there are two modes of pooling: maximum pooling and average pooling. The maximum pooling mode calculates the maximum value among the units in the feature map, and the average pooling mode calculates the average value of the units. In the pooling layer, when M feature maps of the l layer are the input, M feature maps are the output, as shown in the following formula:

$$x^{l+1}_j = f(\beta_j \downarrow \text{down}(x^l_j) + b_j)$$  \hspace{1cm} (6)

where $x_j$ represents the jth input feature map, $x^{l+1}_j$ represents the jth output feature map, $\beta$ and $b$ represent the multiplicative and additive biases corresponding to the jth filter, respectively, $f$ represents the activation function, and “down” represents the subsampling function.

2.2.3. Deconvolution Layer. The deconvolution layer recovers the number of features extracted through the convolution and pooling layers to the resolution of the original feature. It is advantageous in that it can be used to perform upsampling at the same time as the training process, and it can be used to minimize the loss resulting from feature resampling.\(^{17}\)

2.2.4. Training. After obtaining the reconstructed $\hat{X}$ from the output layer, learning is performed so that the reconstruction errors for $X$ and $\hat{X}$ diminish, while the network parameters are refreshed through the backpropagation method, as shown in the following equation:

$$\min_l l_{\text{CNN}} = \frac{1}{2n} \sum_{i=1}^{n} ||x_i - \hat{x}_i||^2$$  \hspace{1cm} (7)

2.2.5. Gradient-Weighted Class Activation Mapping. Grad-CAM is a method that enables the visual explanation of CNN-based prediction results using the gradient information that flows into the last convolutional layer of a CNN. The CNNs used in image analysis comprise a feature extraction part that stacks convolutional and pooling layers over multiple layers and an identification part that receives the feature quantity output and matches it with a class label to perform supervised learning. The identification component typically comprises a fully connected multilayer neural network, and the final layer is used to convert the feature quantity into a probability score for each class.

Grad-CAM is used to identify image locations with a significant effect on the probability score for each class by averaging the changes (derivative coefficients) that occur in the probability scores when an insignificant change is applied to an image location.

First, by applying the formula presented below, the gradient $\partial y_c / \partial A^l_k$ of the intensity $A^l_k$ at the $(i, j)$ pixel of the kth convolutional feature map is calculated using the probability score $y_c$ of class $c$. By averaging these values for all pixels, the weighting factor $\alpha_k$ for the kth filter of class $c$ can be computed. A larger $\alpha_k$ value indicates the increased importance of the feature map $A^l_k$ for class $c$, as follows:

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y_c}{\partial A^l_k}$$  \hspace{1cm} (8)

where Z indicates the number of channels. A heat map of a size that is similar to that of the convolutional feature map is generated by calculating the weighted average of $k$ filters using the calculated value of $\alpha_k^c$ after which the output of the ReLU function is obtained.

$$L_{\text{Grad-CAM}} = \text{ReLU} \left( \sum_k \alpha_k^c A_k \right)$$  \hspace{1cm} (9)

Overlaying onto the input data is possible by resizing.

2.3. Proposed DDD. A CNN can be used to assess the correlation between adjacent elements of the input tensor. However, accurate feature extraction is not possible for unordered process data, even when convolution is performed using a general $3 \times 3$ filter. Accordingly, in this study, only the temporal characteristics of each process variable are initially extracted through the CNN, meaning that the order of process variables does not matter. The DAE is then connected to extract the nonlinearity between process variables. Figure 2 shows an outline of DDD.

To evaluate the process dynamics, the components of the input sample are converted into $m \times (n + 1)$ ($m$ represents the number of input variables, and $n$ represents the number of time delay variables). For temporal feature extraction, the sample is first input into a hidden layer comprising a convolutional layer and pooling layer. Afterward, multidimensional data are converted into one-dimensional data via a fully connected layer. The one-dimensional data are then input into the hidden layer of the DAE to realize the connection between the CNN and the DAE. The number of neurons in the middle layer of the DAE is compressed such that it is at least smaller than $m$ in the input layer. The data are then reconstructed through the decoder and the deconvolution layer. For the loss function $L_x$, the model is trained; therefore, the reconstruction errors at the input layer level are insignificant.

$$L = \frac{1}{m} \sum_{i=1}^{m} || \hat{x}_i - x_i ||^2$$  \hspace{1cm} (10)

In DDD, it is necessary to select the number of convolutional layers, the number of filters, the number of
hidden layers in the AE, and the number of neurons in each hidden layer. The temporal midpoints of the training data used in this study are regarded as the validation data, and the final model was constructed using a combination of hyper-parameters with the smallest L of the temporal midpoints.

DDD detects process faults using two statistics, namely, as $T^2$ statistics and Q statistics, similar to the PCA-based MSCP method. The $T^2$ statistic is calculated using the square of the distance obtained from the origin of the standardized number of the hidden layer neurons, as follows:

$$T^2 = \sum_{i=1}^{d} \left( \frac{t_i}{\sigma_i} \right)^2$$

(11)

where $d$ represents the number of neurons in the middle layer, $t_i$ represents the value of the $i$th neuron, and $\sigma_i$ represents the standard deviation of the $i$th neuron. The Q statistic is obtained from the reconstruction error between the input and output layers, as follows:

$$Q = \|x_i - \hat{x}_i\|^2$$

(12)

Each threshold $\tau_x$ or $\tau_q$ was set to a value containing 99.7% of the total $T^2$ and Q values calculated using the training data. 99.7% is a value that is based on the $3\sigma$ method. The resulting model was used to determine whether the new data were abnormal. If the $T^2$ ($T_{\text{test}}^2$) and Q statistics obtained by inputting the new data into the model ($Q_{\text{test}}$) satisfy either $T_{\text{test}}^2 > \tau_x$ or $Q_{\text{test}} > \tau_q$, the new data are considered abnormal, and the other data are considered normal. The process variables related to such abnormalities are searched when an abnormal condition is detected through the monitoring process. For the $T^2$ statistic, the weight of the $i$th input variable in the $i$th neuron of the hidden layer is represented using $w_{ji}$, and the contribution of the $i$th input variable to the $T^2$ statistic is defined as follows:

$$C_{T^2,i} = \sum_{c=1}^{d} \left( \frac{t_c}{\sigma_c} \right)^2 |w_{ci}|^2$$

(13)

The contribution of the $i$th input variable of the Q statistic is defined as follows:

$$C_{Q,i} = (x_i - \hat{x}_i)^2$$

(14)

3. RESULTS AND DISCUSSION

The TEP dataset was used to verify the effectiveness of DDD. Eastman Chemical Company developed the TEP dataset to mimic an actual industrial process, and this dataset has been used to evaluate the performance of various methods for process control and monitoring. The TEP dataset comprises five main units, namely, a reactor, stripper, condenser, recycle compressor, and separator, with a total of eight components (A through H). The liquid products, G and H, and the by-product, F, are generated from the gaseous reactants A, C, D, and E through chemical reactions. The process is described in detail in the study by Downs and Vogel. The TEP dataset incorporates a total of 52 variables, which include 22 process variables, 11 instrumental variables, and 19 component analysis result variables. In this study, only 22 process measurement variables were used because the process variables are affected because of manipulation. Each process variable employed herein is listed in Table S1. The values for these process variables were measured every 3 min. The training data comprised 1500 min of normal data (500 samples), and the test data comprised 2880 min of data (960 samples) in which 21 types of process faults listed in Table S2 occurred. These datasets and control structures are similar to those previously reported in the literature. In each of 21 types of test data, a process fault occurred after 480 min (160 samples). The models are constructed using the training data, which include only the normal data.

To consider the process dynamics, the samples inputted into DDD were transformed into an $m \times (n + 1)$ matrix, where $m$ represents the number of input variables, and $n$ represents the number of time delay variables. In this study, $m$ was set as 22 for the number of input variables, and $n$ was set as 22, based on the study by Krizhevsky et al. Data were preprocessed by range-scaling at a range of 0–1 for each process variable.

The false negative rate (FNR) (%) and false alarm rate (FAR) (%) were used to evaluate the performance of process fault detection using DDD, as shown below:

$$\text{FNR} = \frac{FP}{TP + FN} \times 100$$

(15)

$$\text{FAR} = \frac{FP}{FP + TN} \times 100$$

(16)

where TP represents the number of samples that are normal when the model is also normal, FN represents the number of samples that are normal when the model is abnormal, and TN represents samples that are actually abnormal when the model is normal. FP represents the number of samples considered normal by the model when the samples are actually abnormal, FNR represents the proportion of classes that are considered abnormal among the actual normal samples, and FAR represents the proportion of classes that are considered normal among the samples that are actually abnormal. Performance improves with a decrease in either FNR or FAR. The capability to detect process faults was determined to be inferior to random estimation if both the FNR and FAR were ≥50%.

The DAE and CNN were used as comparison methods. Table 1 shows the hyperparameters required for each method.

| method | hyperparameter |
|--------|---------------|
| DAE    | $l_{AE}$, $s$ |
| CNN    | $l_{conv}$, $f$ |
| DDD    | $l_{conv}$, $f$, $l_{AE}$, $s$ |

$l_{AE}$ represents the number of hidden layers, $s e R^{l_{AE}}$ represents the rate of reduction from the number of neurons in the previous layer, and $s$ is required only for determining the $l_{AE}$ quantity. $l_{conv}$ represents the number of convolutional layers, and $f e R^{l_{conv}}$ represents the number of filters. $f$ is required only for determining the $l_{conv}$ quantity. Table 2 displays the candidates for each hyperparameter, and Table 3 shows the

| hyperparameter | candidates |
|----------------|------------|
| $l_{AE}$       | 1, 2, 3, 4 |
| $s$            | 1/2, 1/3   |
| $l_{conv}$     | 1, 2, 3, 4 |
| $f$            | 8, 16, 32, 64 |
A combination of hyperparameters with the minimum loss function using the median data from the training dataset. A model using the hyperparameter values listed in Table 3 was constructed, and 21 process faults were detected. Table 4 shows the results of abnormality detection using DAE, CNN, and DDD. In this study, methods that exceeded 50% in terms of either FNR or FAR were not used in performance comparison. Regarding process faults 3, 4, 9, 15, and 16, it was confirmed that no process fault was detected because FAR, FNR, or both exceeded 50% using all methods. DDD exhibited the most favorable FAR among seven process faults, followed by the DAE and the CNN with six and four process faults, respectively. The CNN demonstrated the most favorable FNR among nine process faults, followed by DDD with eight process faults. Figure 3 shows the delay time from 480 min after an abnormality occurs until the point the abnormality is detected. The delay time was confirmed to be insignificant for DDD entirely. The average delay times for DAE, CNN, and DDD were 22.9, 41.7, and 22.4 min, respectively, and DDD exhibited the highest average speed in the detection of process faults.

Table 3. Optimized Hyperparameter Values for Each Method

| method | \( l_{AE} \) | \( s \) | \( l_{conv} \) | \( f \) |
|--------|------------|------|-------------|------|
| DAE    | 4          | 1/2, 1/3, 1/3, 1/3 | 64, 64, 64, 64 |
| CNN    | 3          | 1/3, 1/3, 1/3, 1/3 | 8, 8, 16, 16 |
| DDD    | 3          | 1/3, 1/3, 1/3, 1/3 | 8, 8, 16, 16 |

Table 4. FNR and FAR Results for Each Method in 21 Process Faults

| Process Fault | DAE FNR | DAE FAR | CNN FNR | CNN FAR | DDD FNR | DDD FAR |
|---------------|---------|---------|---------|---------|---------|---------|
| 1             | 15.6    | 0.4     | 4.0     | 0.4     | 24.4    | 0.4     |
| 2             | 8.1     | 1.4     | 4.4     | 1.8     | 4.4     | 1.5     |
| 3             | 44.4    | 51.9    | 26.9    | 54.8    | 10.0    | 72.5    |
| 4             | 11.3    | 73.6    | 4.4     | 76.8    | 9.4     | 72.8    |
| 5             | 11.3    | 48.9    | 4.4     | 56.5    | 9.4     | 54.3    |
| 6             | 6.3     | 0.1     | 1.3     | 0.0     | 7.5     | 0.1     |
| 7             | 12.5    | 32.4    | 8.1     | 37.4    | 6.3     | 35.9    |
| 8             | 19.4    | 0.9     | 3.8     | 0.0     | 2.5     | 1.3     |
| 9             | 48.1    | 52.9    | 33.1    | 57.1    | 31.3    | 63.5    |
| 10            | 11.9    | 14.8    | 16.3    | 14.1    | 9.4     | 11.8    |
| 11            | 17.5    | 1.4     | 20.0    | 0.6     | 24.4    | 0.4     |
| 12            | 25.0    | 0.0     | 17.5    | 0.0     | 7.5     | 0.0     |
| 13            | 8.1     | 4.8     | 1.3     | 5.3     | 6.9     | 3.9     |
| 14            | 18.8    | 0.1     | 8.1     | 0.0     | 21.3    | 0.1     |
| 15            | 6.9     | 61.6    | 1.3     | 60.9    | 9.4     | 65.4    |
| 16            | 71.3    | 10.5    | 61.3    | 11.5    | 63.8    | 13.8    |
| 17            | 24.4    | 2.3     | 19.4    | 2.5     | 25.6    | 1.1     |
| 18            | 11.9    | 6.4     | 13.8    | 7.4     | 8.1     | 6.0     |
| 19            | 8.1     | 3.8     | 3.1     | 1.8     | 1.9     | 1.5     |
| 20            | 10.6    | 12.4    | 0.0     | 13.6    | 2.5     | 13.9    |
| 21            | 26.9    | 27.8    | 27.5    | 41.5    | 10.6    | 29.9    |

Figure 3. Expected fault detection delay: (a) DAE, (b) CNN, and (c) DDD.

Figure 4. Time plot of each statistic for each method in process fault 2. Horizontal straight lines indicate the thresholds, and values in the x-axis indicate the fault detection times. (a) DAE, (b) CNN, and (c) DDD.

The horizontal axis represents the time at which an abnormality is detected using each statistic. According to Figures 4−7, the time plots of the Q statistics for DAE, CNN, and DDD are similar. By contrast, the CNN cannot calculate the T^2 statistic, the DAE indicated subthreshold values in all cases, and neither could detect process faults. Despite this result, it was confirmed (see Figures 4−6) that DDD could accurately detect process faults with respect to the T^2 statistic. For process fault 13, as shown in Figure 6, process faults could be detected earlier compared with using the Q statistic. For process fault 19, as shown in Figure 7, DDD could detect process faults accurately based on the Q statistic. Therefore, it was confirmed that DDD can be used to detect abnormalities accurately.
Subsequently, the process variables related to the abnormalities were identified. For the DDD, the diagnoses of process faults 1, 2, 6, and 13 are outlined in Figures 8–11, respectively. The contribution of each process variable to the process fault can be calculated using the $T^2$ and $Q$ statistics. It was confirmed that DDD can be used to diagnose abnormalities of process variables using the $T^2$ and $Q$ statistics, which cannot otherwise be conducted using conventional deep learning-based techniques. Process variable 1 showed that DDD contributed highly to the $Q$ statistic in process fault 1, which is presented in Figure 8, and an abnormality in the supply flow rate of raw material A was identified. DDD’s $T^2$ statistic further contributed to the process fault detection of process variable 20, thereby suggesting that the compressor

Figure 5. Time plot of each statistic for each method in process fault 6. Horizontal straight lines indicate the thresholds, and values in the x-axis indicate the fault detection times. (a) DAE, (b) CNN, and (c) DDD.

Figure 6. Time plot of each statistic for each method in process fault 13. Horizontal straight lines indicate the thresholds, and values in the x-axis indicate the fault detection times. (a) DAE, (b) CNN, and (c) DDD.
failure is related to the feed rate of raw material A. Based on process fault 2, which is shown in Figure 9, DDD was used to successfully diagnose the presence of an error in the purging of process variable 10. This can be expected to cause abnormalities in the composition of the product. According to process fault 6, as shown in Figure 10, process variables 7, 13, and 16 have larger contributions than other process variables, and abnormalities occur in the reactor, separator, and stripper. However, according to Table S2, this diagnostic result is considered different from the actual cause of the abnormality. It can be concluded that only the $T^2$ statistic of DDD indicates that the supply flow rate of raw material A with respect to process variable 1 is abnormal, and thus, it can contribute to the identification of the root cause of process fault 6. Regarding process fault 13, which is presented in Figure 11, the abnormal pressure levels in the reactor, separator, and
stripper were diagnosed because of the drift in the reaction rate constant. Moreover, DDD was used to successfully determine that there was a substantial contribution by the flow rate of raw material D. Therefore, DDD can be used to increase the information used to identify the causes of process faults by digitizing the degree of influence on the process faults resulting from the high-dimensional feature quantities expressed through deep learning, and it can contribute to the detection of the causes of process faults that cannot be confirmed using conventional methods.

4. CONCLUSIONS

In this study, a deep convolutional neural network with deconvolution and a deep autoencoder (DDD) was proposed for the construction of an MSPC-based deep neural network that assesses the process dynamics and the nonlinearity between process variables. DDD can be used to detect and diagnose process faults through the constructed neural network. Based on the CNN and DAE, DDD can be used to effectively represent the relationship between process variables hidden in process data while simultaneously accounting for the dynamic characteristics and nonlinearity of process variables. By calculating the Q and T² statistics using DDD, it is possible to detect process faults based on these factors, and the T² and Q statistics can be used to digitize the information related to the process variables that contribute to a specific abnormality.

A case study using the TEP dataset was conducted to verify the effectiveness of DDD. DDD can be used to determine the contributions of various process variables to each process fault quantitatively. Overall, compared with conventional process fault detection methods, DDD demonstrates enhanced performance, and its implementation successfully increases the number of determining factors used for identifying the causes of process faults through its ability to effectively present the process variables involved in process faults. Because tensorial data can be analyzed in chemical and biological manufacturing processes, the tensorial data that consider both the process variables and the process dynamics can be analyzed effectively using DDD as future research. However, it should be noted that DDD has limitations in that process data in normal states are required to construct process fault detection and diagnosis models. It is expected that the proposed approach can improve the efficiency of process control and management in chemical plants and industrial facilities through the detection and diagnosis of process faults.

ASSOCIATED CONTENT

Supporting Information
The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acsomega.1c06607.

Process variables and types of process faults in the TEP dataset (PDF)

AUTHOR INFORMATION

Corresponding Author
Hiromasa Kaneko — Department of Applied Chemistry, School of Science and Technology, Meiji University, Kawasaki, Kanagawa 214-8571, Japan; orcid.org/0000-0001-8367-6476; Email: hkaneko@meiji.ac.jp

Author
Yasuhiro Kanno — Department of Applied Chemistry, School of Science and Technology, Meiji University, Kawasaki, Kanagawa 214-8571, Japan

Complete contact information is available at: https://pubs.acs.org/10.1021/acsomega.1c06607

Notes
The authors declare no competing financial interest. The data that support the findings of this study are available in the work of ref 19.

ACKNOWLEDGMENTS

This study was supported by the Grant-in-Aid for Scientific Research (KAKENHI) (grant number 19K15352) from the Japan Society for the Promotion of Science.

REFERENCES

(1) Venkatasubramanian, V.; Rengaswamy, R.; Yin, K.; Kavuri, S. N. A review of process fault detection and diagnosis: Part I: Quantitative model-based methods. Comput. Chem. Eng. 2013, 27, 293–311.
(2) Deng, X.; Tian, X.; Chen, S.; Harris, C. J. Deep learning based nonlinear principal component analysis for industrial process fault detection. In 2017 International Joint Conference on Neural Networks 2017, 1237–1243.
(3) Wise, B. M.; Ricker, N. L.; Veltkamp, D. F.; Kowalski, B. R. A theoretical basis for the use of principle component models for monitoring multivariate processes. Process Control Qual. 1990, 1, 41–51.
(4) Kano, M.; Tanaka, S.; Hasebe, S.; Hashimoto, I.; Ohno, H. Monitoring independent components for fault detection. AIChE J. 2003, 49, 969–976.
(5) MacGregor, J. F.; Jaeckle, C.; Kiparisssides, C.; Koutoudi, M. Process monitoring and diagnosis by multiblock PLS methods. AIChE J. 1994, 40, 826–838.
(6) Watanabe, K.; Matsuura, I.; Abe, M.; Kubota, M.; Himmelblau, D. M. Incipient fault diagnosis of chemical processes via artificial neural networks. AIChE J. 1989, 35, 1803–1812.
(7) Peng, Y.; Chen, X.; Ye, Q.; Jiao, J. Fault detection and classification in chemical processes using NMFS and structural SVMs. Can. J. Chem. Eng. 2014, 92, 1016–1023.
(8) Hinton, G. E.; Salakhutdino, R. R. Reducing the dimensionality of data with neural networks. Science 2006, 313, 504–507.
(9) Dong, C.; Loy, C. C.; He, K.; Tang, X. Image Super-Resolution Using Deep Convolutional Networks. IEEE Trans. Pattern Anal. Mach. Int. 2016, 38, 295–307.
(10) Wu, H.; Zhao, J. Deep convolutional neural network model based chemical process fault diagnosis. Comput. Chem. Eng. 2018, 115, 185–197.
(11) Wen, L.; Li, X.; Gao, L.; Zhang, Y. A new convolutional neural network-based data-driven fault diagnosis method. IEEE Trans. Ind. Electron. 2018, 65, 3990–3998.
(12) Xu, G.; Liu, M.; Jiang, Z.; Söflker, D.; Shen, W. Bearing fault diagnosis method based on deep convolutional neural network and random forest ensemble learning. Sensors 2019, 19, 1018.
(13) Jiang, G.; Xie, P.; He, H.; Yan, J. Wind turbine fault detection using a denoising autoencoder with temporal information. IEEE/ASME Trans. Mechatron. 2018, 23, 89–100.
(14) Selvaraju, R. R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; Batra, D. Grad-CAM: Visual explanations from deep networks via gradient-based localization. IEEE Int. Conf. Comput. Vision 2017, 618–626.
(15) Hansen, M. F.; Smith, M. L.; Smith, L. N.; Smith, N.; Saltar, M. G.; Baxter, E. M.; Farish, M.; Grieve, B. Towards on-farm pig face recognition using convolutional neural networks. Comput. Ind. 2018, 98, 145–152.
(16) Yang, C.; Rangarajan, A.; Ranka, S. Visual explanations from deep 3D convolutional neural networks for Alzheimer’s disease classification. AMIA 2018, 2018, 1571−1580.
(17) Zhao, H.; Liu, H.; Hu, W.; Yan, X. Anomaly detection and fault analysis of wind turbine components based on deep learning network. Renewable Energy 2018, 127, 825−834.
(18) Downs, J. J.; Vogel, E. F. A plant-wide industrial process control problem. Comput. Chem. Eng. 1993, 17, 245−255.
(19) Russell, E. L.; Chiang, L. H.; Braatz, R. D. Fault detection in industrial processes using canonical variate analysis and dynamic principal component analysis. Chemom. Intell. Lab. Syst. 2000, 51, 81−93.
(20) Krizhevsky, A.; Sutskever, I.; Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks. Commun. ACM 2017, 60, 84−90.
(21) Sun, W.; Braatz, R. D. Opportunities in tensorial data analytics for chemical and biological manufacturing processes. Comput. Chem. Eng. 2020, 143, 107099.