Improving Convolutional Neural Networks for Fault Diagnosis by Assimilating Global Features

Saif S. S. Al-Wahaibi and Qiugang Lu

Abstract—Deep learning techniques have become prominent in modern fault diagnosis for complex processes. In particular, convolutional neural networks (CNNs) have shown an appealing capacity to deal with multivariate time-series data when converted into images. However, existing CNN techniques mainly focus on capturing local or multi-scale features from input images. A deep CNN is often required to indirectly extract global features, which are critical to describing the images converted from multivariate dynamical data. This paper proposes a novel local-global scale CNN (LGS-CNN) architecture that directly accounts for both local and global features for fault diagnosis. Specifically, the local features are acquired by traditional local kernels, whereas global features are extracted using one-dimensional tall and fat kernels that span the entire height and width of the image. Both local and global features are then merged for classification using fully-connected layers. The proposed LGS-CNN is validated on the benchmark Tennessee Eastman process dataset. Comparison with traditional CNN shows that the proposed LGS-CNN can greatly improve the fault diagnosis performance without significantly increasing the model complexity. This is attributed to the much wider local receptive field created by the LGS-CNN than that by CNN. The proposed LGS-CNN can also outperform artificial neural networks and fisher discriminant analysis in FD on the same dataset.

I. INTRODUCTION

Over the last decade, deep learning (DL) has attracted increasing attention for fault diagnosis (FD). Primarily, the strength of DL lies in its ability to utilize the extensive data present in industrial systems to establish complex models for distinguishing anomalies, diagnosing faults, and forecasting without needing much prior knowledge [1]. Among various DL methods for FD, convolutional neural networks (CNNs) have shown great promise due to their efficiency in capturing spatiotemporal correlations and reduced trainable parameters from weight sharing [2].

Originally developed for image classification, CNNs entail neural networks consisting of convolutions with local kernels and pooling operations to extract features from images [2], [3]. They have also been used in FD to handle time-series data. Janssens et al. [4] made one of the earliest attempts at using CNNs for FD. The authors highlighted the capability of CNNs to learn new features from input images converted from time-series data to classify faults in rotating machinery better. Further developments on CNN for FD can be referred to in [5]–[10]. Note that different from images in computer vision, the images converted from time-series data often possess strong non-localized features. To this end, kernels of different sizes, i.e., multi-scale CNN, have been used in [5] to cover local receptive fields (LRF) with varying resolutions to improve the learned features. Other techniques, such as global average pooling, have been employed in [10], [11] to maintain the integrity of information pertaining to global correlations. However, these approaches either can only directly capture broader local features (e.g., multi-scale CNN) or lack learnable parameters in acquiring global correlations (e.g., global average pooling). Thus, they often need to construct deep networks to capture global features that are crucial in multivariate time-series data for FD [12]. In addition, most research studies mentioned above are mainly concerned with one-dimensional (1D) or low-dimensional time-series data such as the wheel bearing data [6]. Research on extending CNN for FD for high-dimensional multivariate time-series data, e.g., those obtained from chemical processes, still remains limited.

One exemplary work is reported in [7], where deep CNNs are constructed to diagnose faults from the Tennessee Eastman process (TEP). Gramian angular field is used in [8] to convert multi-dimensional data into multi-channel two-dimensional (2D) images to apply CNN for FD. Nevertheless, these works still cannot directly extract global features from the multivariate time-series data or, equivalently, the formed 2D images. Instead, they also rely on constructing deep CNNs to expand the LRF to the entire image for capturing global correlations. As a result, the number of trainable parameters can easily go beyond several million, causing significant training complexity [7], [8]. Hence, there is a pressing demand for developing a novel CNN-based FD framework that adequately extracts global spatiotemporal correlations while maintaining a reasonable number of parameters for multivariate time-series datasets.

This paper proposes a novel local-global scale CNN (LGS-CNN) framework for FD for complex dynamical processes. The proposed framework converts multivariate time-series data into images and collects both global and local features simultaneously to classify faults. Local correlations are captured using typical local square kernels, whereas global correlations are integrated using 1D tall (temporal) and fat (spatial) kernels that span the entire height and width of the image. The spatial and temporal global features extracted from the tall and fat kernels are then cohered together to acquire global spatiotemporal patterns in the images. Such global spatiotemporal features are then concatenated with local features extracted with the typical square kernels to
merge the information prior to FD. While many of the reviewed CNN-based approaches attempt to capture global correlations, LGS-CNN introduces a novel method to do so without the need for excessively deep architectures containing many trainable parameters. The developed LGS-CNN is validated with the TEP data, and simulation results show that LGS-CNN’s incorporation of global features can greatly enhance FD performance compared with CNN with a modest model complexity increase.

This paper is organized as follows. Section II presents fundamentals about traditional CNNs. The proposed LGS-CNN architecture is elaborated in Section III, followed by a case study of FD for TEP in Section IV. The conclusions are given in Section V.

II. PRELIMINARIES

In this section, we briefly introduce the main components in a typical CNN including convolutional, pooling, and fully-connected (FC) layers. In addition, batch normalization (BN) is introduced to mitigate internal covariance shift issues.

A. Convolutional Layers

In a convolutional layer, a kernel filter slides across an input feature map where an affine transformation is conducted at every slide location such that:

$$ C^l_i = b^l_j + \sum_{l=1}^{l-1} X^{l-1}_{i,j} \ast K^l_{i,j}, \quad j = 1, 2, \ldots, I_l, $$

where $K^l_{i,j} \in \mathbb{R}^{k \times \eta}$ is the kernel of size $k \times \eta$ in layer $l \in \{1, 2, \ldots, L\}$ and channel $i \in \{1, 2, \ldots, I_{l-1}\}$, $X^{l-1}_{i,j} \in \mathbb{R}^{n \times m}$ is the input feature map of size $n \times m$ to layer $l$. $L$ is the number of layers and $I_l$ is the number of channels in the $l$-th layer. $C^l_i$ is the $j$-th output map in layer $l$ after the convolution and $b^l_j$ is the bias. The symbol $\ast$ represents the convolution operation. An activation function, such as the rectified linear unit (ReLU), is usually applied to $C^l_i$ to add non-linearity. Graphically, the first green feature map in Fig. 1 illustrates a $3 \times 3$ convolution. The square patch of size $3 \times 3$ in the input image represents the LRF of the dark green neuron output in the first feature map in the top branch. Thereby, an LRF can be thought of as the “field of view” incorporated in calculating a new feature through the convolutional operation.

B. Pooling Layers

Pooling operations after convolutional layers often act as a sub-sampling step to reduce dimensionality while preserving information. Specifically, in a localized group of activations on a feature map, pooling summarizes their responses through either averaging or maximizing operations [13]. We use max pooling on each local $s \times s$ region of the input feature map $X^{l-1}_{i,j} \in \mathbb{R}^{n \times m}$, and the resultant new feature map is shown to be [11]

$$ P^l_i = \max_{r=1}^{s} X^{l-1}_{i \cdot r}, $$

where $P^l_i$ is the output of the max pooling operation, and $S$ is the total number of $s \times s$ regions in $X^{l-1}_{i,j}$.

C. Fully-Connected Layers

After all convolutional and pooling layers, the obtained feature maps represent the main features learned by a CNN from an input image. These maps are then flattened into a vector and passed through FC layers for classification or regression. Specifically, the output of the $l$-th FC layer is calculated using

$$ d^l_z = b^l_z + x^{l-1} \ast \omega^l_z, \quad z = 1, 2, \ldots, Z, $$

where $d^l_z$ is the output of the $z$-th neuron, $Z$ is total number of neurons in the $l$-th layer, $b^l_z$ is the bias, $x^{l-1} \in \mathbb{R}^{\zeta}$ is the activation vector from the previous layer that contains $\zeta$ neurons, $\omega^l_z \in \mathbb{R}^{\zeta \times \zeta}$ is the weight vector associated with neuron $z$, and $\cdot$ represents the dot product. To add non-linearity, an activation function is applied to $d^l_z$.

D. Batch Normalization

BN accelerates CNN learning by reducing the effects of internal covariance shift [14]. Layers experience these effects in the learning process when previous layers update their weights and biases resulting in a need to continuously adapt to these changes, and therefore, hindering the learning process. In a 2D-CNN, these effects are mitigated by normalizing the activations from a preceding layer:

$$ \tilde{X}^{l-1}_{i,j} = \frac{X^{l-1}_{i,j} - \mathbb{E}[X^{l-1}_{i,j}]}{\sqrt{\text{VAR}[X^{l-1}_{i,j}]}} $$

where $\tilde{X}^{l-1}_{i,j} \in \mathbb{R}^{n \times m}$ is the $i$-th channel in the $(l-1)$-th layer, $\mathbb{E}[-]$ is the expectation over the training batch and all pixel locations, and $\text{VAR}[-]$ is the variance. Then, representation is restored to the layer by the affine computation [14]

$$ Y^l_i = \tilde{X}^{l-1}_{i,j} a^l_j + \beta^l_j, $$

where $Y^l_i \in \mathbb{R}^{n \times m}$ is the BN output for channel $i$ in layer $l$, and $a^l_j$ and $\beta^l_j$ are learnable parameters for each channel.

III. METHODOLOGY

The proposed LGS-CNN shown in Fig. 1 consists of multiscale convolutions to extract both local and global features. In addition, we use $1 \times 1$ convolution [6], max pooling, and strided convolution [15] for dimension reduction with minimum information loss.

A. Local Correlations

The top branch in LGS-CNN (green color in Fig. 1) shows the usage of traditional $3 \times 3$ kernels to extract local features from the input image. Note that here we apply BN steps before the ReLU activations. Further, padding is added to ensure that the output has the same dimension as the input image. In addition, to reduce dimensions, $1 \times 1$ convolution is conducted to squeeze the number of channels after the convolution. Traditionally, $3 \times 3$ kernels usually capture a small LRF region [3], [9]. The overall mapping
Convolution
20 x 50 Input Image
1 x 1 Convolution
1 x 50 Convolution
20 x 1 Convolution
1 x 1 Convolution
3 x 3 Convolution
1 x 1 Convolution
Channel Concatenation
3 x 3 Convolution
2 x 2 Pooling
3 x 3 Convolution (Stride = 3)
Fully-Connected Layer
Output Layer
20 x 50 Input Image
1 x 50 Convolution
1 x 1 Convolution
3 x 3 Convolution
1 x 50 Convolution
20 x 1 Convolution
1 x 1 Convolution
3 x 3 Convolution
1 x 1 Convolution

Fig. 1. The proposed LGS-CNN architecture to extract both local and global features.

from the input image to the extracted feature maps after $1 \times 1$ convolutions is expressed as:

$$\Psi = f_{\theta_L}(X^0) \in \mathbb{R}^{c_L \times n \times m},$$

where $\Psi$ is the feature maps extracted from the top branch, $f_{\theta_L}(\cdot)$ represents all operations from input images to the extracted feature maps in the top branch, with a collection of trainable parameters into $\theta_L$, and $c_L$ (subscript $L$ represents “local”) is the channel number in the extracted feature maps.

### B. Global Correlations

The novelty in the proposed architecture resides in the bottom branch of the LGS-CNN model (blue, gold, and red blocks in Fig. 1). To capture global correlations, we propose to use 1D tall (gold) and fat (blue) kernels that encompass the entire width and height of the images so that

$$\Phi = f_{\theta_u}(X^0) \in \mathbb{R}^{c_u \times n \times 1}, \quad \Omega = f_{\theta_g}(X^0) \in \mathbb{R}^{c_g \times 1 \times m},$$

where $\Phi$ and $\Omega$ are the obtained feature maps, $f_{\theta_u}(\cdot)$ and $f_{\theta_g}(\cdot)$ represent the width-wise and height-wise convolutions (see (1)) parameterized by $\theta_u$ and $\theta_g$, respectively, and $c'_G$ stands for the channel number after these convolutions (subscript $G$ represents “global”). To capture the global spatiotemporal coherent features, the acquired feature maps are multiplied together across each channel to give new features in the form of 2D maps:

$$Y = \Phi \otimes \Omega \in \mathbb{R}^{c'_G \times n \times m},$$

where $\otimes$ is the outer product of feature vectors obtained after the width-wise and height-wise convolutions, and $Y$ represents the ultimate feature maps obtained from such multiplication across all $c'_G$ channels. For clarity, Fig. 2 presents a calculation graph of the aforementioned operations on a dummy $3 \times 3$ matrix. Similar to the top branch, $1 \times 1$ convolution takes place after obtaining $Y$ in (8) to shrink the channel number from $c'_G$ to $c_G$. In addition, to reduce the risk of divergence or a vanishing gradient due to the multiplication step, BN is applied after it and before the $1 \times 1$ convolution. Inclusive of all the operations detailed, the following description summarizes the overall mapping in the bottom branch:

$$\Pi = f_{\theta_b}(X^0) \in \mathbb{R}^{c_l \times n \times m},$$

where $\Pi$ is the bottom branch’s feature maps, $f_{\theta_b}(\cdot)$ entails all the operations in this branch with BN and ReLU implementations, and $\theta_b$ are all the parameters involved.

### C. Max Pooling and Strided Convolution

After the multi-scale convolutions, feature maps from different branches are concatenated, channel-wise, such that

$$\Xi = [\Psi \Pi] \in \mathbb{R}^{(c_l+w_G) \times n \times m},$$

where $\Xi$ is the combined output from the multi-scale convolutions. Subsequently, a $3 \times 3$ convolution layer is conducted to integrate all features, followed by a max pooling step to reduce the dimension. The latter operation employs a $2 \times 2$ pooling with a stride of 2 to reduce the size of each dimension by half. Interestingly, max pooling does not just serve as a dimensionality reduction step, it also adds non-linearity and
regularizes [11]. To further reduce dimensionality while at the same time maintaining information, a strided convolution takes place after max pooling. Principally, it is just a typical square convolution with a stride larger than one [16]. Here we use a $3 \times 3$ convolution with a stride of 3. As such, the dimension reduction steps can be summarized as

$$\Gamma = f_{\theta_m}(\Xi) \in \mathbb{R}^{(c_L+c_G) \times \kappa \times \varepsilon},$$

where $\Gamma$ are the resultant feature maps of dimensions $\kappa \times \varepsilon$ with $c_L+c_G$ channels, which will be flattened and then fed to the classification step. $f_{\theta_m}(\cdot)$ represents the entire mapping parameterized by $\theta_m$.

D. Fully-Connected Layers

Lastly, FC layers with one hidden layer conclude the proposed network architecture. To aid with classification, softmax activation is applied to the output vector of the FC layers such that

$$\hat{y} = f_{\theta_{fc}}(\text{vec}(\Gamma)) \in \mathbb{R}^{C \times 1},$$

where $\hat{y}$ is the network output after softmax function with $C$ representing the number of classes. $f_{\theta_{fc}}(\cdot)$ is the overall mapping, and $\theta_{fc}$ stacks all learnable parameters in the FC layers. The largest entry in $\hat{y}$ indicates the class that the input image shall be classified into.

E. Comparison with Related Works

In the proposed method, the 1D tall and fat kernels are crucial in extracting global features from input images. Using non-square or vector kernels for enhancing feature representation has been reported in the literature. For instance, vector kernels are employed in [17]. However, the LRFs are expanded via a $2 \times 2$ convolution preceding the vector kernels. In our work, the multiplication step in (8) can allow variables that are far away to be correlated to extract coherent global features from the image, thus, expanding the LRF. Similarly, in [18], vector kernels are also utilized for augmenting semantic segmentation. Nonetheless, the authors apply 1D fat and tall kernels (but with small size) in series to replace traditional square kernels. In contrast, our method applies the height-wise tall and width-wise fat kernels in parallel. Another related work is reported in [19], where traditional square kernels are replaced by tall and fat kernels for popular deep CNN models such as AlexNet and VGG. With such vector kernels, the number of trainable parameters can be reduced with improved classification performance. However, the work in [19] does not study global feature extraction specifically.

IV. SIMULATION

In this section, we use the benchmark TEP data [20] to assess the performance of the proposed LGS-CNN. This dataset contains 41 measured and 11 manipulated variables from a simulation of a chemical process consisting of a reactor, condenser, compressor, separator, and stripper [21]. Specifically, the dataset contains 20 different types of faulty data. In this paper, we utilize 47 simulations for each of the 20 faults (40 for training and 7 for testing). The data are sampled every 3 minutes for 25 hours for training and 48 hours for testing. Thus, each training and testing simulation contains 500 samples and 960 samples, respectively. Therefore, a total of 400,000 samples for the training dataset and 134,400 samples for the testing datasets are selected.

A. Preprocessing

In this stage, variables representing compressor recycle and stripper stream valves are dropped because they remain constant for some simulations [7]. In addition, data samples collected before the faults were introduced to each simulation are removed [21]. As a result, only 384,000 training and 112,000 testing samples, with each sample containing 50 variables, are used for training and testing.

To apply the proposed method, the raw multivariate time-series data shall be first converted into images. To do so, every 20 samples from the TEP data are saved as a 2D (gray) image, of dimension $20 \times 50$, with one channel. Furthermore, both training and testing images are normalized according to the mean and standard deviation of the training images (see (4)). Eventually, a training dataset of 19,200 images and a testing dataset of 5,600 images are obtained for this case study.

B. Correlation Map

To study the cross-correlation among process variables, Fig. 3 shows the correlation coefficient heat map across all 50 process variables with the data from Fault 2. It is seen that strong correlations are widely scattered, indicating that different columns which are far apart in the obtained images above can possess strong correlations. Thus, both local and global correlations shall be considered when dealing with the images converted from the time-series data.

C. Training and Validation

The cross-entropy criterion is used to guide the learning process by minimizing the following objective function:

$$J(\Theta) = \frac{1}{M} \left[ \sum_{m=1}^{M} \sum_{c=1}^{C} y_m \log P(\hat{y}_m = c | x_m, \Theta) \right],$$

Fig. 3. Correlation coefficients of process variables in the TEP.
TABLE I
DIFFERENT MODEL STRUCTURES DEVELOPED FOR THE LGS-CNN.

| Model | 1 | 2 | 3 |
|-------|---|---|---|
| Input(1, 20, 50) | C((3, 3), 16) | C((3, 3), 32) | C((3, 3), 64) |
| Branch 1 | BN | BN | BN |
| C((1, 1), 8) | C((1, 1), 16) | C((1, 1), 32) |
| Branch 2 | C((20, 1, 50)) | C((20, 1, 32)) | C((20, 1, 64)) |
| Branch 3 | C((20, 1, 50)) | C((1, 50), 32) | C((1, 50), 64) |
| Global Branch | Vector Multiplication(Branch 2, Branch 3) | BN | BN |
| C((1, 1), 8) | C((1, 1), 16) | C((1, 1), 32) |

Main Concatenation(Branch 1, Global Branch)
- C((3, 3), 64) |
- BN |
- P((2, 2), 2) |
- C((3, 3), 64) |
- s = 3 |
- BN |
- FC(16000, 20) |
- FC(2304, 300) |
- FC(4608, 300) |
- FC(300, 20) |
- Parameters 321,668 |
- 719,952 |
- 1,509,552 |

D. Ablation Experiments

To assess the FD performance, the fault diagnosis ratio (FDR) is defined as the ratio between correct classification and the total sample number for a fault class:

\[
FDR = \frac{TP}{TP + FN},
\]

where \(TP\) is the number of true positives, and \(FN\) is the number of false negatives. For this study, the performance of the proposed LGS-CNN, with different model structures as in Table I, is assessed against that of traditional CNNs of the same structures but without the global feature extraction as in Table II. Fig. 4 compares the averaged FDRs from CNN and LGS-CNN for the three models. Table III details the FDRs for 20 faults.

and the total sample number for a fault class:

where \(TP\) is the number of true positives, and \(FN\) is the number of false negatives. For this study, the performance of the proposed LGS-CNN, with different model structures as in Table I, is assessed against that of traditional CNNs of the same structures but without the global feature extraction as in Table II. Fig. 4 compares the averaged FDRs from CNN and LGS-CNN for the three models. It is seen that LGS-CNN clearly outperforms CNN consistently. The only penalty for the LGS-CNN is a small increase in the number of learnable parameters. However, LGS-CNN Model 1 performs better (0.872 vs. 0.859) than CNN Model 2, with fewer parameters (321,668 vs. 716,528). Thus, with LGS-CNN, the model complexity can be significantly reduced to yield comparable performance to CNN, indicating the significance of acquiring global features. In addition, a comparison with an artificial neural network (ANN) with 1,573,396 parameters and Fisher discriminant analysis (FDA) shows that LGS-CNN Model 3 can perform better (0.865 vs. 0.540 vs. 0.906, respectively).

To further compare the performance between LGS-CNN, CNN, ANN, and FDA, we use Table III to detail the FDRs for all 20 faults. It is clear that the LGS-CNN model presents better performance across more faults. In particular, the LGS-CNN model performs considerably better for Faults 3, 9, and...
15, which are notoriously difficult to diagnose [7], [8].

All these results clearly indicate the superior performance of our proposed LGS-CNN models in FD for the TEP dataset. Such observations can be attributed to the large LRFs acquired by employing the 1D height-wise tall and width-wise fat kernels. To further illustrate, Fig. 5 demonstrates the LRFs captured by CNN and LGS-CNN Models 3. It is observed that the tall and fat kernels from LGS-CNN produce an LRF that covers a much broader area in the input image than that from CNN. In other words, the LGS-CNN can easily capture global features without building unnecessarily deep networks. Such deep networks typically demand many trainable parameters that could lead to overfitting. It is suspected that ANN performance is hindered due to overfitting given that they can capture global features. On the other hand, FDA is a linear dimension reduction technique that can be poor in capturing complex non-linear relationship common in complex chemical processes.

V. CONCLUSION

This paper proposed a novel LGS-CNN model for the FD of complex dynamic processes. In the proposed model architecture, local features from images obtained from multivariate time-series data are captured by traditional square kernels, whereas global features are captured by tall and fat kernels that cover the entire height and width of the image, respectively. Both local and global features are then concatenated in the fully-connected layer for FD. The proposed method is validated on a benchmark TEP dataset. Simulation results show that LGS-CNN performs better in FD tasks compared with traditional CNN and ANN. Moreover, the LGS-CNN can employ a much simpler structure than CNN to yield a similar level of FD rates. This observation lies in the much wider LRF created by LGS-CNN than that of CNN, which benefits obtaining global features. Future work aims at testing the proposed LGS-CNN on actual industrial data. The proposed technique can also be migrated to other image processing and computer vision tasks due to the simplicity of adding the global feature extraction mechanism.

ACKNOWLEDGMENT

S. Al-Wahaibi acknowledges the Distinguished Graduate Student Assistantship and Q. Lu acknowledges the startup funds from the Texas Tech University.

REFERENCES

[1] T. Wuest, D. Weimer, C. Irgens, and K.-D. Thoben, “Machine learning in manufacturing: Advantages, challenges, and applications,” Production & Manufacturing Research, vol. 4, no. 1, pp. 23–45, 2016.
[2] Y. LeCun, Y. Bengio, et al., “Convolutional networks for images, speech, and time series,” The Handbook of Brain Theory and Neural Networks, vol. 3361, no. 10, p. 1995, 1995.
[3] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, “1D convolutional neural networks and applications: A survey,” Mechanical Systems and Signal Processing, vol. 151, p. 107398, 2021.
[4] O. Janssens, V. Slavkovikj, B. Vervisch, K. Stockman, M. Loccutt, S. Verstockt, R. Van de Walle, and S. Van Hoecke, “Convolutional neural network based fault detection for rotating machinery,” Journal of Sound and Vibration, vol. 377, pp. 331–345, 2016.
[5] W. Huang, J. Cheng, Y. Yang, and G. Guo, “An improved deep convolutional neural network with multi-scale information for bearing fault diagnosis,” Neurocomputing, vol. 359, pp. 77–92, 2019.
[6] T. Jin, C. Yan, C. Chen, Z. Yang, H. Tian, and S. Wang, “Light neural network with fewer parameters based on CNN for fault diagnosis of rotating machinery,” Measurement, vol. 181, p. 109639, 2021.
[7] H. Wu and J. Zhao, “Deep convolutional neural network model based chemical process fault diagnosis,” Computers & Chemical Engineering, vol. 115, pp. 435–437, 2017.
[8] S. Sun and J. Ren, “GASF-MSNN: A new fault diagnosis model for spatiotemporal information extraction,” Industrial & Engineering Chemistry Research, vol. 60, no. 17, pp. 6235–6248, 2021.
[9] J. Yan, T. Liu, X. Ye, Q. Jing, and Y. Dai, “Rotating machinery fault diagnosis based on a novel lightweight convolutional neural network,” Plos One, vol. 16, no. 8, p. e0256287, 2021.
[10] H. Luo, L. Bo, C. Peng, and D. Hou, “Fault diagnosis for high-speed train axle-box bearing using simplified shallow information fusion convolutional neural network,” Sensors, vol. 20, no. 17, p. 4930, 2020.
[11] R. Nirthika, S. Manivannan, A. Ramanan, and R. Wang, “Pooling in convolutional neural networks for medical image analysis: A survey and an empirical study,” Neural Computing and Applications, pp. 1–27, 2022.
[12] Q. Lu, B. Jiang, and E. Harinath, “Fault diagnosis in industrial processes by maximizing pairwise kullback–leibler divergence,” IEEE Transactions on Control Systems Technology, vol. 29, no. 2, pp. 780–785, 2019.
[13] Y.-L. Boureau, J. Ponce, and Y. LeCun, “A theoretical analysis of feature pooling in visual recognition,” in Proceedings of the 27th International Conference on Machine Learning (ICML-10), pp. 111–118, 2010.
[14] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in International Conference on Machine Learning, pp. 448–456, PMLR, 2015.
[15] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9, 2015.
[16] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, “Striving for simplicity: The all convolutional net,” arXiv preprint arXiv:1412.6806, 2014.
[17] J. Chen, Z. Lu, J.-H. Xue, and Q. Liao, “Xception: Extremely separated convolution,” arXiv preprint arXiv:2002.12046, 2020.
[18] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun, “Large kernel matters–improve semantic segmentation by global convolutional network,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 435–4361, 2017.
[19] J. Ou and Y. Li, “Vector-kernel convolutional neural networks,” Neurocomputing, vol. 330, pp. 253–258, 2019.
[20] C. A. Rieth, B. D. Amsel, R. Tran, and M. B. Cook, “Additional Tennessee Eastman Process Simulation Data for Anomaly Detection Evaluation,” 2017.
[21] L. H. Chang, E. L. Russell, and R. D. Braatz, Fault Detection and Diagnosis in Industrial Systems. Springer Science & Business Media, 2000.