A survey of Autoencoder and Convolutional Neural Network Based Methods for Fault Diagnosis

Chengfeng Rao
School of Northeastern University, Shenyang, 110000, China
2200893@stu.neu.edu.cn

Abstract. Deep learning technology, with its exceptional nonlinear feature extraction capability, has been gradually accepted and widely adopted by the industry. This paper reviews the fault diagnosis techniques utilizing deep learning, emphasizing an in-depth analysis of two main deep learning models that are extensively applied in the current fault diagnosis domain: Autoencoder (AE) and Convolutional Neural Network (CNN). In addition, we also discuss the specific application cases of both algorithms, and introduce their specific principles of implementation. Through these in-depth analysis, it is intended to provide readers with the application overview and technical progress of deep learning technology in the field of fault diagnosis, so as to offer reference and enlightenment for related research.

Keywords: Bearing fault diagnosis; Deep learning; Autoencoder; Convolutional Neural Network.

1. Introduction

As the manufacturing sector progresses, there has been an escalating demand for the reliability of mechanical equipment. Predictive Health Management (PHM) technology has emerged as a novel solution, garnering widespread attention. Rotational machinery is a category of equipment that relies on rotary motion to fulfill its functions, having extensive applications across a wide range of industries. In the mechanical system, bearings are designed to support equipment components and minimize the coefficient of friction to ensure the precision of rotation. Therefore, bearings are indispensable for ensuring the smooth operation of equipment and improving production efficiency. However, due to a myriad of factors, including fatigue, wear, deformation, corrosion and fracture, bearings may fail, leading to equipment downtime and production loss. Consequently, the diagnosis of rolling bearing failures has become an essential strategy in safeguarding equipment safety and boosting production efficiencies. Typically, intelligent diagnostic approaches encompass three main stages: data acquisition, feature extraction and model construction.

During the data acquisition phase, a variety of sensors are deployed to monitor and record the operational status information of the bearings. For example, vibration sensors are employed to gather the vibration signals of the bearings. Acoustic emission sensors are used to capture the acoustic emission signals. Infrared thermal imaging sensors can be utilized to collect temperature signals from the bearings.

In the feature extraction stage, the deep learning model identifies and abstracts key information by autonomously learning the complex hierarchical structures of data, thereby providing high-quality feature representations for subsequent fault diagnosis processes. In order to enhance the saliency of data features and facilitate the learning efficiency of the network, data analysis is often performed in advance, such as time domain, frequency domain and time-frequency domain. Various signal processing techniques can be adopted to facilitate this, including Short-Time Fourier Transform, Continuous Wavelet Transform, Variational Mode Decomposition, Wigner-Ville Distribution, Symmetrical Dot Pattern, Empirical Mode Decomposition and its advanced methods like Ensemble EMD and Complete Ensemble EMD.

The construction of the model involves designing an algorithm capable of distinguishing and differentiating information. In the traditional control field, model construction primarily relies on the expert's prior knowledge to ensure model performance. However, in the field of deep learning, it reduces the dependence on experts, as deep models can automatically extract features. On the one
hand, traditional machine learning methods usually require well-designed feature engineering. In contrast, deep learning models can autonomously learn features, reducing the need for feature engineering. On the other hand, though traditional machine learning algorithms can also process data, their computational complexity often scales with the size of the data. Deep learning algorithms can train on large datasets effectively by breaking them into smaller batches, thus enabling them to efficiently handle extensive datasets. Commonly used network types are Artificial Neural Network[20], Support Vector Machine [21], Kernel Extreme Learning Machine[22], K-Nearest Neighbour (KNN)[23], and so on.

Deep learning, as a branch of machine learning, realizes the automatic extraction and learning of data features by constructing a deep network structure. The key principles of most deep learning networks today are as follows:

1. The composition of the network layers is a combination of linear and nonlinear functions[24]. In the early stages, most neural network models and machine learning models predominantly used linear layers to build models, restricting them to capturing only the linear relationships in data and failing to grasp more complex non-linear relationships. As research progressed, the limitations of linear models became apparent, prompting attempts to incorporate non-linear elements to augment the model's expressive capacity. This was mainly achieved by adding non-linear activation functions after linear layers. Widely used activation functions include Sigmoid, Rectified Linear Unit, and Gaussian Error Linear Unit[25].

2. Use the chain rule (also known as backpropagation) to adjust parameters in the entire network at once. The BP algorithm was proposed by Geoffrey Hinton in 1988[26]. It involved two phases, forward propagation and backward propagation. In modern deep learning models, Stochastic Gradient Descent[27] and Adaptive moment estimation [28] algorithms are often used in supervised learning to calculate gradient information for updating model parameters.

As deep learning research deepens and iterates, it has become a powerful tool for handling unstructured data and complex tasks. Among numerous deep learning networks, Autoencoder (AE) and Convolutional Neural Network (CNN) have attracted wide attention field of intelligent diagnosis due to their simple network architecture, fast running speed and excellent feature extraction capabilities. Therefore, this paper will systematically review the research results of AE and CNN network in the field of rolling bearing fault diagnosis.

The structure of this paper is arranged as follows: the second section elaborates the AE network and its application; the third section takes a look at CNNs and their related implementation.

2. Autoencoder and its Variants

With the widespread application of deep learning techniques in the field of intelligent diagnosis, deep neural network models based on AE have gradually garnered increasing attention. This section will introduce the standard AE network and its variants, including Denoising Autoencoder (DAE) and Stacked Autoencoder (SAE).

2.1 Standard AE

AE is an artificial neural network widely used in unsupervised learning, which was first proposed by Rumelhart et al[26]. Its primary objective is to learn how to extract high-dimensional feature representations of input information, achieving effective compression of the information. As shown in Fig.1, an AE usually consists of several fully connected neural networks layers, which are categorized based on their functions into the input layer, the hidden layer, and the output layer. The encoder network comprises the input and hidden layers, while the decoder network consists of the hidden and output layers[29]. The objective function of an AE is designed to minimize the discrepancy between the input and its reconstructed output, aiming to enable the AE to capture the fundamental characteristics and structure of the data under normal and faulty conditions, so as to
effectively identifying anomalous signals. The forward inference process of an AE consists of two steps: the Encoder and the Decoder.

![AE Structure](image)

**Fig. 1 AE Structure**

The main role of the encoder is to map the input information into a unified feature space for feature classification, and help to refine the information and reduce the information dimension. Given an input vector \( x_i(x = [x_1, x_2, ..., x_i]) \), the Encoder will map \( x_i \) to a latent feature vector \( h_i(h = [h_1, h_2, ..., h_i]) \), as shown below:

\[
h_i = f(W_e x_i + b_e)
\]

(1)

where \( W_e \) represents the weight matrix of the Encoder network, \( b_e \) is the bias vector of the Encoder network. \( f(\cdot) \) is an activation function, introduced to incorporate non-linear factors.

The decoder aims to reconstruct data that is as close as possible to the original input from the highly abstracted representation distilled by the encoder. It progressively decompresses this information through a layer-by-layer network, and ultimately mapping it back to the original data space. By using \( h_i \) as input for the Decoder network, its output \( \hat{x}_i \) can be obtained as shown below:

\[
\hat{x}_i = W_d h_i + b_d
\]

(2)

where \( W_d \) represents the weight matrix of the Decoder network, \( b_d \) is the bias vector of the Decoder network. The vector \( \hat{x} \) is the reconstructed output of the Decoder network.

During model training, AE often uses the cross-entropy loss function \( L_{\text{cross-entropy}}(W_e, W_d, b_e, b_d) \) to guide the optimization process of parameters. This loss function measures the divergence between the model's output and the actual values. By minimizing this loss value, the model is encouraged to learn an efficient representation of the data.

\[
L_{\text{cross-entropy}}(W_e, W_d, b_e, b_d) = -\frac{1}{m} \sum_{i=1}^{m} x_i \log(\hat{x}_i) + (1 - x_i) \log(1 - \hat{x}_i)
\]

(3)

Alternatively, the Mean Squared Error (MSE) loss can be chosen as the loss function. The formula \( L_{\text{MSE}}(W_e, W_d, b_e, b_d) \) for its computation is as follows:

\[
L_{\text{MSE}}(W_e, W_d, b_e, b_d) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{x}_i - x_i)^2
\]

(4)

Since the Decoder in an Autoencoder architecture reconstructs information using latent vectors, it is generally considered that the Encoder accomplishes feature extraction and dimension reduction of the input, while the latent vectors are regarded as a compressed representation of the input.
2.2 DAE

The DAE is proposed to mitigate overfitting during training, thereby enhancing its robustness and generalization capabilities. The central idea of this method is to guide the model to reconstruct the original input from information that has been artificially corrupted with noise. Gaussian noise and salt-and-pepper noise are typically chosen as the types of noise to be added. Fig. 2 shows the flowchart of the forward inference of DAE. The formula for adding noise to the original input is given below.

\[ \bar{x}_i = x_i + n_i \]  

where \( x_i \) represents the original input. \( \bar{x}_i \) denotes the input after noise addition. \( n_i \) denotes the added random noise.

2.3 SAE

The SAE is proposed to address the limitation of shallow machine learning models in capturing complex information in data. Compared to the standard AE, the SAE enhances its feature extraction capabilities by stacking multiple layers of AE, enabling the network to learn features at different levels. As shown in Fig. 3, the SAE is a deep neural network composed of multiple AE. Specifically, the output of the hidden layers from each preceding AE serves as input for the subsequent layer's AE, and these AEs are stacked together to form the entire SAE network.

For an SAE with \( n \) layers, the forward inference of the Encoder part is shown as follows:

\[ h^k = f(W^e_k h^{k-1} + b^e_k), k \in [1, n] \]  

where \( k \) denotes the \( k^{th} \) layer of the AE. \( h^k \) represents the output of the \( k^{th} \) hidden layer. \( h^0 \) indicates the original input \( x \). \( W^e_k \) represents the weights of the Encoder network at the \( k^{th} \) layer.

The forward inference of the Decoder part is shown as follows:

\[ \hat{x}^k = f(W^d_{k-1} h^{k-1} + b^d_{k-1}), k \in [1, n] \]
where $W_{d}^{n-(k-1)}$ represents the Decoder weight of the $(n-(k-1))^{th}$ layer. $h^{k-1}$ is the hidden layer output of the $(k-1)^{th}$ layer. $b_{d}^{n-(k-1)}$ denotes the bias of the $(n-(k-1))^{th}$ layer Decoder.

Since the SAE network belongs to the deep network, it is easy to encounter the problem of difficult convergence during training. Hinton et al. proposed a training strategy called greedy layer-wise training[31] to alleviate the gradient problem during training. This training process is divided into two stages: the pre-training phase and the fine-tuning phase.

The pre-training stage involves unsupervised training for each AE layer to ensure that the hidden layers of each AE reach a local optimum state. Specifically, the training starts with the first AE. Firstly, its decoding part is completed. Then, the distance between the original input and the reconstructed value can be obtained by using a loss function such as NCE, which can guide the update of model parameters. The output from the hidden layers of the AE is used as the input for the next AE, which is trained by using similar methods as before. This process continues sequentially until all AEs have been trained, and then these AEs are connected in order.

The global fine-tuning takes the parameters of the SAE network obtained from the unsupervised training are used as the initial values for the model. Then, based on this, the entire network undergoes fine-tuning until convergence.

### 2.4 Application of Autoencoder

To further enhance the performance of network models and address the complexities of real-world industrial environments, improved methods based on the AE and their variants have been proposed.

Since the training of AE adopts the self-supervised learning method, its ability to distinguish class features was limited. To address this, Ref.[32] proposed a feature distance stack autoencoder (FD-RAE) to guide the model to better distinguish the feature information between different categories by introducing feature distance information. Specifically, it was necessary to calculate the average feature distance $d_i$ between samples of the same class, as well as the distance $d_{ij}$ between samples of different classes.

$$d_i = \frac{1}{N} \sum_{k=1}^{N} \| h_k - c_i(h) \|^2 \quad (i = 1, 2, \ldots, m) \quad (8)$$

$$d_{ij} = \frac{N_i + N_j}{(m-1)N} \| c_i(h) - c_j(h) \|^2 \quad (1 \leq i \leq m-1, i < j \leq m) \quad (9)$$

where the $c_i(h) = \frac{1}{N_i} \sum_{k=1}^{N_i} h_k$ represented the feature center of the $i^{th}$ category of samples. $N_i$ denoted the number of samples in the $i^{th}$ class ($N = \sum_{i=1}^{m} N_i$). $\frac{N_i + N_j}{(m-1)N}$ represented the distance weighting coefficient between the $i^{th}$ and $j^{th}$ categories.

After obtaining $d_i$ and $d_{ij}$, it aimed to reduce the feature distance within samples of the same class while increasing the feature distance between samples of different classes. The loss $L_{FD-RAE}$ was as follows:

$$p(h) = \sum_{i=1}^{m} d_i - \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} d_{ij} + \xi \| h \|^2 \quad (10)$$

$$L_{FD-RAE} = -\frac{1}{N} \sum_{k=1}^{N} x_k \cdot \log \hat{x}_k + \lambda \sum_{l=1}^{L} \| w^{(l)} \|^2 + \eta \sum_{l=1}^{L} p^{(l)} \quad (11)$$

where $x_k$ and $\hat{x}$ represented the expected value and the predicted value of the $k^{th}$ sample. $w^{(l)}$ was the weight matrix of the $l^{th}$ layer. $\xi$ was the weight decay hyperparameter. $\eta$ was the weight parameter. $p^{(l)} = p(h^{(l)})$ was the distance penalty term.
Compared to networks relying on a single sensor, adopting a multi-sensor approach could provide richer information for deep learning networks. Ref.[33] proposed a fusion network that extracted features from individual sensors using multiple independent autoencoders. These features were then merged and further processed for feature extraction, achieving a more holistic integration of information.

Also in order to fully leverage data from multiple sensors and further enhance model performance, Ref.[34] introduced a multi-modal variational autoencoder (MMVAE). It utilized VAE as the feature extraction network for each modality. Subsequently, different modalities were set as reference in turn. The reference feature was used as the target to guide the clustering learning of features from other modalities. Compared with GMM and KNN, MMVAE demonstrated superior fault diagnosis performance, and its accuracy reached more than 96%.

Compared with the multi-sensor system that only uses the same kind of sensors as discussed in [33], the multi-modal information could describe the operation state of the equipment more comprehensively. Ref.[35] proposed a deep coupling autoencoder (DCAE), which consisted of three stacked blocks: a gaussian bernoulli restricted boltzmann machine (GBRBM), a restricted boltzmann machine (RBM), and a coupling autoencoder (CAE). Specifically, the combination of GBRBM and RBM networks was used to extract high-level feature representations, and then the correlation between the two modalities was learned by CAE. The structure of CAE was shown in Fig.4, which guided the model to capture the common features between different modalities by calculating the similarity of the hidden layer outputs between two AEs. The definition of similarity measure was as follows.

$$S(x_v, y_v; \theta_v, \theta_a) = \|f_v(x_v; \theta_v) - f_a(x_v; \theta_a)\|^2$$  \hspace{1cm} (12)

where $x_v$ and $x_a$ represented the original vibration and acoustic signals. $f_v(x_v; \theta_v)$ and $f_a(x_v; \theta_a)$ were the vibration modality and the acoustic modality. The model parameters denoted as $\theta_v$ and $\theta_a$.

In terms of model optimization, aiming at the problem that the sigmoid activation function was easy to be saturated due to the KL divergence used in the standard SAE model, Ref.[36] proposed an improved SAE based on convolutional shortcuts and domain fusion strategy (ISAE-CSDF). This approach employed residual blocks as a replacement for KL divergence, thus mitigating the vanishing gradient problem and enhancing model performance.

Besides, the acquisition of faulty samples was usually challenging, leading to an imbalanced distribution of sample categories in datasets. This issue could severely impact the network's performance. To address this problem, in terms of network design, Ref.[37] proposed an adaptive sparse contractive autoencoder (ASCAE). The sparsity constraint was added to the CAE, and homotopy regularization was used to adjust the sparse coefficients in order to enhance the generalization performance.
What’s more, in order to address the issue of class imbalance in datasets, Ref.[38] proposed the CVAEGAN-SM to generate samples and balance the dataset. Experimental results showed that, compared to previous methods, the problem of overfitting was alleviated, and the model's generalization capability was enhanced.

3. **Convolutional Neural Network**

3.1 1-D CNN and 2-D CNN

![Fig. 5 CNN Architecture](image)

CNN was first introduced in the 1990s[39]. The advantage of CNN is that it has the property of translation invariance, which enables CNN to automatically learn spatial features without being affected by the position of the spatial object. As shown in Fig.5, a standard CNN is composed of three main types of layers: convolutional layers, pooling layers, and fully connected layers. The convolutional layer is responsible for performing convolution operations on the input data to extract features, while the pooling layer is used to reduce the spatial dimension of the feature map, thereby reducing the network parameters and the computational complexity. The main role of the fully connected layer is to integrate and map the local features extracted by the previous layers to achieve the decision output of a specific task, such as classification or regression. By sequentially combining convolutional and pooling layers into multiple convolutional blocks, and then stacking these blocks with several fully connected layers, a complete CNN is constructed.

Convolutional layers, as a pivotal component of CNNs, are constructed based on three hyperparameters: the size of the convolutional kernel (or filter), stride, and padding strategy. The size of the convolution kernel determines the network's ability to perceive the local region size in the input data. Small convolution kernels can capture more fine-grained features, whereas large kernels facilitate the acquisition of more extensive contextual information. The stride determines the interval when the convolution kernel slides over the input data. A large stride can reduce the spatial dimension of the output feature map but some detail information may be lost. Padding is employed to add additional zero values around the boundaries of the input data, enabling the convolution operation to cover areas at the edges of the image, thus controlling the spatial dimension of the output feature map.

In the specific application scenario of bearing fault diagnosis, 1-D (one-dimension) convolutional layers and 2-D (two-dimension) convolutional layers are mainly utilized. 1-D convolutional layers are typically used to process time series data, such as sensor signals. And 2-D convolutional layers are more suitable for analyzing 2-D grid-type data.

![Fig. 6 1-D Conv Kernel](image)

The algorithm of convolution lies in the element-wise multiplication and summation operation performed by the convolutional kernel (or filter) on the input feature map, which serves to generate the value of each element in the output feature map. As shown in Fig.6, for an 1-D convolutional layer, given a 1-D input vector \( x = [x_1, x_2, \cdots, x_l] \), its process can be described by the following formula:
where $y(i)$ represents the value of the $i$th element in the output sequence. $M$ denotes the size of the one-dimensional convolution kernel. $W$ represents the weight matrix of the convolution kernel.

\[
y(i) = (w * x)[i] = \sum_{m=0}^{M-1} x[i] \cdot h[i-m]
\]  

(13)

Fig. 7 2-D Convolutional Layer

At this point, as shown in Fig.7, 2-D convolution calculation formula can be extended as follows.

\[
y(i, j) = (W * X)[i, j] = \sum_{m=0}^{M-1} \sum_{n=0}^{M-1} X(i-m, j-n)W(m,n)
\]  

(14)

where $X$ is the 2-D input matrix, $W$ is the convolution kernel matrix, and $Y$ is the convolution output matrix. $i, j$ denote the row index and column index of the output matrix.

Pooling layers help reduce sensitivity to subtle changes and remove redundant information to prevent overfitting. There are primarily two types of pooling layers, the first of which is the max pooling layer.

\[
y_{m,n}^d = \max_{i \in R_{m,n}} x_i
\]  

(15)

where $R_{m,n}^d$ represents the pooling window. $x_i$ denotes the element values within the window.

The second type is the average pooling layer.

\[
y_{m,n}^d = \frac{1}{|R_{m,n}^d|} \sum_{i \in R_{m,n}} x_i
\]  

(16)

3.2 Application of Convolutional Neural Network

Research related to CNNs is mainly divided into two directions: on the one hand, some researches are devoted to the development of new network architectures that are able to directly process raw data as input; on the other hand, another stream of researchs focus on data preprocessing and transformation techniques, aiming to optimize the data to better fit the existing network architecture. These trends reflect the diverse strategies adopted when designing and deploying CNNs.

3.2.1 Data Adaptation to Networks

In the research field of bearing fault detection, the collected signal is usually an 1-D time series, which directly reflects the running status of the bearings. Therefore, how to transform the 1-D signals into a format suitable for 2-D CNN processing has become a research trend.

Ref.[40] proposed a data processing method that converted 1-D time-domain signals into 2-D images, so as to break the input limit of image-based CNN. Specifically, a segment of length $M^2$ was randomly selected from the input original sequence, where $M$ was the length and width of the generated 2-D image. $L(i)$ denoted the value at the $i^{th}$ position of this segment, where $i = 0, \ldots, M^2 - 1$. In this case, the image pixel value $P(j,k)$, where $j, k \in [1,M]$, was calculated as follows:

\[
P(i, k) = \text{round}\left\{\frac{L((j-1)\times M + k) - \text{Min}(L)}{\text{Max}(L) - \text{Min}(L)} \times 255\right\}
\]  

(17)

Although this work proposed a straightforward data processing approach, the continuity of the time series may be destroyed during the conversion process. Furthermore, the spatial relationships between rows in the 2-D image did not exist in the original 1-D sequence, which could lead the model to learn irrelevant or erroneous features.
Similarly, in order to convert 1-D data into 2-D data, Ref. [41] achieved this transformation by concatenating data from multiple sensors row by row to form a two-dimensional matrix, thereby facilitating the conversion from 1-D signals to 2-D data. In this way, it not only effectively overcame the problem of information loss in the work [40], but also enabled the efficient utilization of information from multiple sensors. This facilitated the model in more effectively capturing and studying the mutual relationship and complementary among different sensors, which significantly improved the overall performance of the model. The experimental evidence demonstrated that, in this work, the average accuracy over ten trials was 99.83%, which was 2.25% higher than the average accuracy achieved with a single sensor.

In addition to simply concatenating sensor data, Ref. [42] implemented the conversion of 1-D signals to 2-D images by using vibration signals of sensors to generate shaft orbit images. Subsequently, fault detection was completed through a 2-D CNN. Through data from various turbine sensors, it was proved that the verification accuracy reached 100% for four faults: imbalance, friction, misalignment and oil whirl.

Similarly, in an attempt to convert sensors data into images that could be recognized by a 2-D CNN, Ref. [43] employed the Stockwell transform (S-transform) to describe the energy fluctuations of sensors across different frequencies and time scales. Moreover, in order to mitigate the impact of noise and enhance the edge information of the image, the Sobel filter was applied. The filtered images were called SobelEdge Scalograms. The accuracy, precision, recall and F1 scores obtained by the proposed method in the real world industrial vibration dataset were 99.68%, 99.65%, 99.68% and 99.66%.

3.2.2 Network Adaptation to Data

For the 1-D time series input, compared with the research direction of data catering to the network, exploring network architectures to adapt to the characteristics of the data is also a potential research direction. This necessitates an in-depth analysis of the properties of time series data to design network structures that can fully exploit the available information.

The time series signals measured by sensors showed multi-scale features [44, 45, 46]. However, traditional CNNs were primarily designed for feature extraction at a single scale, which resulted in their lack of ability to extract such multi-scale features. In order to solve this limitation, MSCNN [47] proposed a multi-scale CNN architecture. By downsampling from different time scales, multiple inputs of different coarse-grained were extended, and then a set of features extracted at different scales could be obtained after an 1-D CNN. Finally, by combining this feature set and passing it through a fully connected network, the classification result could be made. Specifically, given a sequence of measurements $x = \{x_1, x_2, \ldots, x_N\}$, where $x_i$ was the value at timestamp $i$. Continuous coarse-grained signals $y_j^{(s)}$ were constructed by averaging the data points in the original signal $x$.

$$
y_j^{(s)} = \frac{1}{s} \sum_{i=(j-1)s+1}^{j} x_i, 1 \leq j \leq \frac{N}{s}$$

where $s$ represented the window length.

In the pursuit of extracting features across multiple scales, instead of dividing the data at multiple scales by [47], Ref. [48] constructed a multiscale feature extraction block utilizing layers with wide convolutional kernels, multiscale convolutional layers, and layers composed of multiple small convolutional kernels. The multiscale convolutional layer, in particular, was designed to fuse features extracted in parallel by convolutional kernels of varying sizes. Furthermore, it also added residual connections to mitigate the problem of gradient vanishing. On the CWRU dataset, this method achieved 99.21% average accuracy, which was 3.13% higher than TICNN [49] and 3.26% higher than WDCNN [50].

In addition, different from single modal methods, Ref. [51] proposed a multi-modal fusion network based on vibration and acoustics. Its main idea was to use 1-D CNNs to extract the features of each mode, and then fused the features of all modalities through a fusion network. Finally, it used the
fusion features to complete the fault diagnosis. Experimental results indicated that by exploiting the complementary information among multiple modalities to a certain extent, this method significantly enhanced the model's resistance to noise interference.

Finally, different from only using CNN to extract features, Ref.[52] proposed to use CBAM[53] to guide the CNN network to extract critical feature information, and use BiLSTM[54] for temporal feature extraction. On the CWRU dataset, the proposed method achieved an average accuracy of over 99.78%, while on the VALENIAN-PT500 measured data, the accuracy reached 99.79%.

4. Conclusion

In this article, we present a overview of two popular deep learning models commonly utilized in the area of fault diagnosis: AE and CNN. Through the detailed introduction and analysis of these algorithms, compared with the traditional machine learning algorithms, which usually require a lot of prior knowledge and human-designed features, deep learning algorithms demonstrate significant potential in solving bearing fault diagnosis tasks.

AE, trained in an unsupervised way, can learn to effectively distinguish and extract feature information from a large number of unlabeled data. This capability is crucial, as acquiring data, especially a large amount of labeled data, is often challenging in real-world scenarios. With simple structure and greedy layer-wise training method, AE is not only easy to construct but also capable to study high-level feature representations automatically from large-scale, unlabeled complex data. In practical application scenarios, the standard AE is often difficult to be competent for complex feature extraction tasks. Therefore, numerous researchers made innovations based on the standard AE to improve the ability and performance of autoencoder variants in feature extraction by adjusting the network architecture or altering the input. These enhancements include stacking multiple AE layers, introducing sparsity constraints, and artificially adding noise, etc., aiming to capture the deep features in the data through more complex and efficient mechanisms.

With the unique structural design of convolutional layer, such as local connection and weight sharing mechanism, and multi-layer to increase the receptive field, CNN shows superior feature extraction capacity compared to traditional fully connected network, while also reduces the complexity of the model. Additionally, the translation invariance of CNN ensures that the model maintains a consistent response to the position changes of the input data. This means that no matter how the input information moves or shifts in space, the CNN is able to recognize and make the same judgment. On the other hand, from the perspective of network adaptation to data, feature extraction and classification are completed by directly inputting 1-D data into the network.

Looking forward to the future, fault diagnosis methods rooted in deep learning can be developed along the following key directions. Firstly, the fusion of multimodal data can bring more comprehensive feature information, significantly enhancing the accuracy and reliability of diagnosis. Besides, in the context of the high cost of industrial data collection, transfer learning shows its great application value and research potential. In addition, through the comprehensive use of various deep learning models, such as Recurrent Neural Network, Long Short-Term memory network and CNN, the advantages of each model can be complementary, and the rich feature information contained in the data can be mined and utilized more efficiently.

References


