Hindawi
Advances in Civil Engineering
Volume 2020, Article ID 6680315, 9 pages
https://doi.org/10.1155/2020/6680315

Research Article

Bearing Defect Classification Algorithm Based on Autoencoder Neural Network

Manhuai Lu[1] **and Yuanxiang Mou**[2]

[1]*College of Mechanical and Electrical Engineering, University of Electronic Science and Technology of China, Zhongshan Institute, Zhongshan 11545, China*
[2]*School of Mechanical and Electrical Engineering, University of Electronic Science and Technology, Chengdu 10614, China*

Correspondence should be addressed to Yuanxiang Mou; 1966764053@qq.com

Received 26 October 2020; Revised 13 November 2020; Accepted 4 December 2020; Published 14 December 2020

Academic Editor: Wei Liu

Copyright © 2020 Manhuai Lu and Yuanxiang Mou. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The postproduction defect classification and detection of bearings still relies on manual detection, which is time-consuming and tedious. To address this, we propose a bearing defect classification network based on an autoencoder to enhance the efficiency and accuracy of bearing defect detection. An improved autoencoder is used to reduce dimension feature extraction and reduce large-scale images to small-scale images through encoder dimensional reduction. Defect classification is completed by feeding the extracted features into a convolutional classification network. Comparative experiments show that the neural network can effectively complete feature selection and substantially improve classification accuracy while avoiding the laborious algorithm of the conventional method.

1. Introduction

Bearing quality is related to the overall performance of a machine, affecting its stable operation and indirectly affecting the quality of its output. Bearing surface defects are a key factor in the life cycle of bearings based on the internet of things [1–5]. Bearing manufacturers attach great importance to the quality of their products and generally inspect them before they leave the factory. A bearing's outer ring surface is more prone to defects than other parts in the assembly and production process, and it has a great impact on a machine's performance. Therefore, bearing inspection focuses on the outer ring surface. Bearing factories presently rely on manual sampling inspection (Figure 1). This method is inefficient, and not all outputs can be tested, which may lead to overlooked defects. Also, due to the influence of experience, working state, and fatigue of inspectors, detection standards cannot be unified. Therefore, it is necessary to design automatic detection equipment.

With high speed, nondestructive characteristics, low noise, and automation capabilities, machine vision systems have found a wide range of applications in product defect inspection. This process is well suited to bearing manufacturing. Inspection requires knowledge of whether defects exist and the types of defects. The probability distribution of types of defects should be determined to address production problems and make improvements.

2. Related Work

Defect-classification algorithms have been developed. Features used as raw data for classification have a direct effect on the results, and their performance depends largely on the form of data and the choice of features.

Belkin [6] proposed a manifold dimensionality reduction method based on the conversion of high-dimensional initial samples to low-dimensional manifold structures to reduce the sample dimensionality and facilitate sample visualization. Sooraksa et al. [7] aimed to detect defects on an air bearing surface. They combined block matrix-based image segmentation technology with a neural network to recognize defects. Çelik and Dülger [8] detected and

Figure 1: Manual sampling inspection of bearings.

classified four commonly occurring defects in fabrics—lack of warp yarn, weft yarn defect, yarn hole contamination, and yarn flow—using an algorithm based on a wavelet transform, image morphology analysis, and binary operations. The defect-classification algorithm was based on a grayscale co-occurrence matrix and feedforward neural networks. It achieved defect-detection accuracy as high as 93.4% and defect-classification accuracy of 96.3%.

Extracted original feature data in image processing are often high-dimensional and contain considerable redundant information. To directly process the original data requires substantial computing resources, so it is often necessary to reduce its dimensionality. Principal component analysis (PCA) is a commonly used unsupervised dimensionality reduction algorithm. Minka [9] proposed Bayesian model selection to estimate the true dimension of data when determining the number of principal components retained by the PCA algorithm and applied the Laplace method after selecting the appropriate parameters to solve the integration problem on the Stiefel framework. This led to better results and a higher processing speed compared to a cross-validation algorithm.

The choice of classification algorithm has a huge impact on classification results. Mien Van and Hee-Jun Kang [10] proposed a wavelet-kernel local linear Fisher discriminant analysis algorithm (WKLFDA) using a wavelet-kernel function for linear Fisher discriminant analysis (LFDA). Particle swarm optimization was used to automatically select the parameters of WKLFDA and to convert the multi-classification problem into binary classification using a one-versus-one strategy. The features of each binary classification process after dimensionality reduction were fed into a single support vector machine (SVM), whose results were combined with a decision fusion mechanism to determine the condition of a bearing. The classification efficiency of this algorithm was better than that of other algorithms, and its classification accuracy could reach 98.80%. Zapata et al. [11] used 12 geometric characteristics to represent the shapes and orientations of weld seam defects and proposed an artificial neural network (ANN) and an adaptive neuro-fuzzy inference system (ANFIS) to classify defects. Through experiments, the correlation coefficient and trust matrix of the ANN and ANFIS were determined, and respective classification accuracies of 78.9% and 82.6% were achieved.

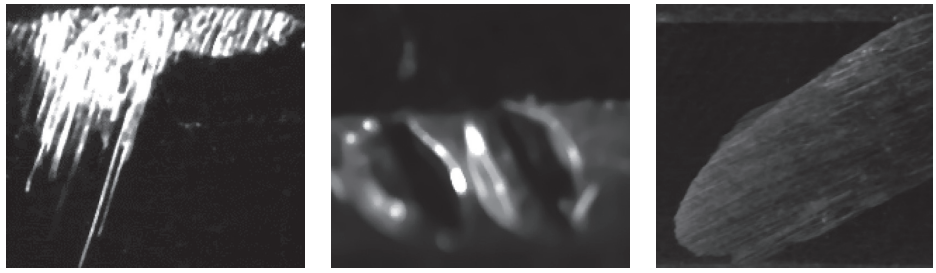
There are many classic network models. The Alex network [12] used dropout technology to improve the overfitting problem. Google network used 1×1 convolution and proposed a concept embedding structure, using average pooling instead of full connection to greatly improve the network depth. It was the champion of ILSVRC detection and classification in 2014. It used the convolution pooling full connection VGG network [13], and the convolution kernel of all convolution layers was 3×3, which greatly improved the expression ability. The Res network [14] introduced a residual structure, which made it easier to optimize a deep network and further deepened it.

The concept of a neural network was proposed in the 1940s when the multi-perceptron (MP) prototype model and Hebb learning rules first appeared. Rosenblatt [15] proposed a perceptron model and laid the foundation for neural networks. Rumelhart [16] proposed a back-propagation algorithm (BP) to solve the problem that perceptron models could only have single layers. Based on the BP algorithm, LeCun [17] proposed a convolutional neural network suitable for deep learning, and Hinton [18] proposed a deep belief network that used a hierarchical initialization method to train deep network parameters, solved the bottleneck of the BP algorithm, and made it possible to increase the number of network layers. Thus, the concept of deep learning was born. A neural network can directly take inputs and output a classification result. We apply neural network technology to classify bearing defects.

3. Methods

3.1. Equilibration and Enhancement of Data. Based on defects that actually occur in bearing production, we collected three types of representative defect samples—abrasions, bruises, and overgrinding defects—on model 6204 deep groove ball bearings, as shown in Figure 2. Of these samples, 308 were of abrasion, 240 bruises, and 247 overgrinding. A network trained on a dataset with significantly different numbers of samples in each category would acquire inconsistent recognition ability of different categories. Thus, we balanced the samples to 240 per category using undersampling, for a total of 720 samples. This order of magnitude was small in terms of the training of a deep-layer neural network. More samples usually give better training results. We increased the number of samples to 4320 by rotating each defect sample by 90°, 180°, and 270° to obtain horizontal and vertical mirror images. Of these samples, 1296 were used as a testing set and the remaining 3024 were used as the training set.

3.2. Preprocessing of Samples. In training a network, the speed of training and robustness of the model are enhanced by using samples of the same size, so it is necessary to normalize the sample size. The size range of the original samples was 50–600, with most samples concentrated



(a) (b) (c)

Figure 2: Three types of bearing defects. (a) Abrasion: wear caused by friction between a bearing and a rough object as it moves on a production line. (b) Bruise: when moving between stations, the bearing transfer device does not fix the bearing, which causes concavity from the collision between the bearing and other objects. (c) Overgrinding: in the machining process, excessive grinding of the bearing is caused by machine vibration or incorrect parameter settings.

between 100 and 300. If the image sizes were decreased too much, some feature information could be lost. If too much of the sample size was retained, then redundant information would increase the computational cost. Considering these two factors, we normalized the sample sizes to 112×112 by linear interpolation.

Three typical filters—Gaussian, median, and mean—were compared experimentally, as shown in Figure 3. Figure 3(a) shows the original image, where brighter areas correspond to defects. There are many irregularly distributed white spots caused by small impurities on the bearing surface or noise generated by electromagnetic interference during signal transmission. The templates of the Gaussian, mean, and median filters were all 7×7. The Gaussian and median filters could smooth out the noise to a certain extent but not completely. The Gaussian filter used different weighting coefficients and achieved better results than the mean filter. None of the filters could preserve the defect textures. Blurred textures can cause an image to lose features that are helpful in classification. The median filter could completely remove noise regions of a certain size and reduce the area of larger noise areas, while retaining the texture characteristics of defects. A comprehensive comparison of the effects of the three filtering algorithms showed that the median filter should be selected for denoising.

3.3. Bearing Defect Classification Network Based on Convolutional Autoencoder

3.3.1. Convolutional Autoencoder. A typical autoencoder [19] is a three-layered unsupervised neural network consisting of input, hidden, and output layers. The input and hidden layers constitute the encoder, and the hidden and output layers constitute the decoder. The network structure is shown in Figure 4.

The functional relationship between the input and output of the encoding process can be expressed as follows:

$$y = S_1(W_1 x + b_1), \tag{1}$$

where $S_1(x)$ is the encoder activation function and W_1 and b_1 are the weight and bias, respectively, of a neuron. The decoding process can be expressed as follows:

$$z = S_2(W_2 y + b_2), \tag{2}$$

where $S_2(x)$ is the activation function and W_2 and b_2 are the weight and bias, respectively, of a decoder neuron. When y has a smaller dimension than x, the autoencoder can be used for dimensionality-reduction feature extraction. The training of the autoencoder generally aims to minimize the reconstruction error, and the loss function is generally chosen as the mean squared error between the reconstruction and the input. This loss function can effectively quantify the difference between the output and input, and its derivative can be obtained as follows:

$$L(X, Z) = \frac{1}{N} \sum_{i=1}^{N} (x_i - z_i)^2. \tag{3}$$

3.3.2. Improved Convolutional Autoencoder Bearing Defect Classification Network. Figure 5 shows the network training process of the model structure designed in this paper. The bearing defect samples were photographs, but conventional fully connected autoencoders are not suitable for image processing. Through the sharing of weighting factors, a convolutional autoencoder can greatly reduce the number of parameters. We adopted a convolutional autoencoder to extract features from the defect samples. The convolution process is shown in Figure 6, and the network layer parameters are listed in Table 1.

Based on the characteristics of the bearing defect samples, we made the following improvements to the autoencoder structure.

(1) Elimination of Pooling. In a conventional convolutional network, convolution is followed by pooling to reduce the dimension of the output and the size of the feature map. Figure 7 shows maximum pooling with a sliding step of 2×2. After pooling, the size of the feature map is reduced by half. To improve the stability of the autoencoder and reduce the number of network layers, the convolution stride was set to two to replace the original pooling operation. The network will automatically learn the appropriate sampling function during training.

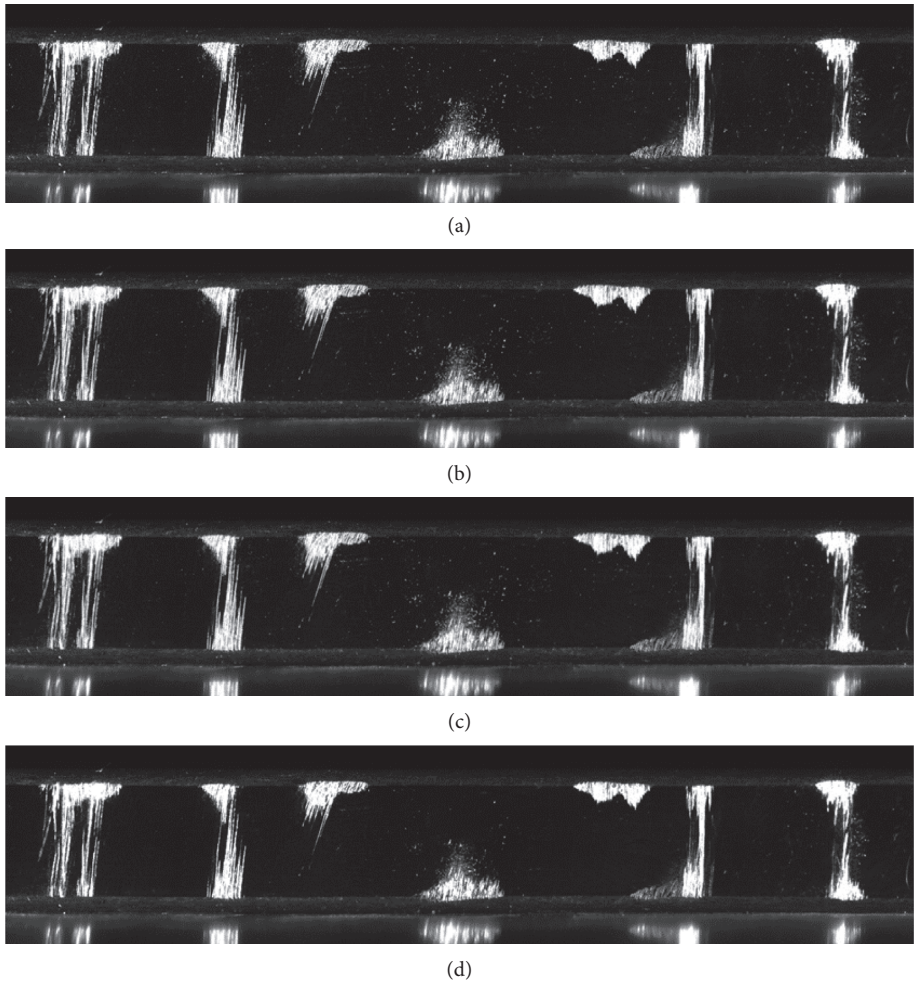


Figure 3: Performance comparison of three filter algorithms: (a) original image; (b) Gaussian filter with $\sigma = 1.55$; (c) mean filter; and (d) median filter.

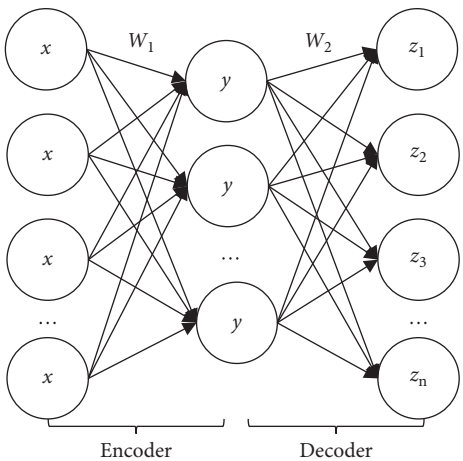


Figure 4: Typical autoencoder mechanism.

(2) Leaky ReLU Activation Function. Rectified linear unit (ReLU) activation functions are used extensively in neural networks. Their positive semiaxis derivative remains unchanged, which facilitates propagation in the gradient direction. However, some information may be lost when the input data to ReLU [12, 13, 20–22] is less than zero. To retain such input data, we use the revised leaky ReLU activation function. This improves the portion that is less than zero based on the ReLU function. Leaky ReLU is expressed as follows:

$$f(x) = \begin{cases} ax, x < 0, \\ x, x \geq 0, \end{cases} \tag{4}$$

where a is the minimum nonzero parameter that can be learned from the directional propagation algorithm. Figure 8 shows the function and its derivative. This function enables the avoidance of the issue of the neuron not being activated in backpropagation of the model.

After dimensionality reduction and feature extraction by the autoencoder, the data are sent to the classification network for classification.

The pooling operation is removed, and the network structure parameters are modified to meet the classification requirements of this project. The classified network structure parameters are shown in Table 2.

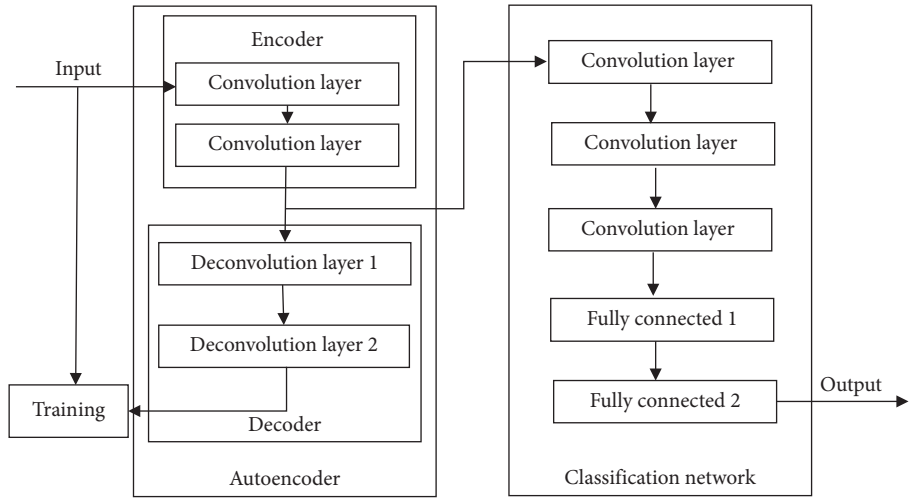


Figure 5: Overall network structure.

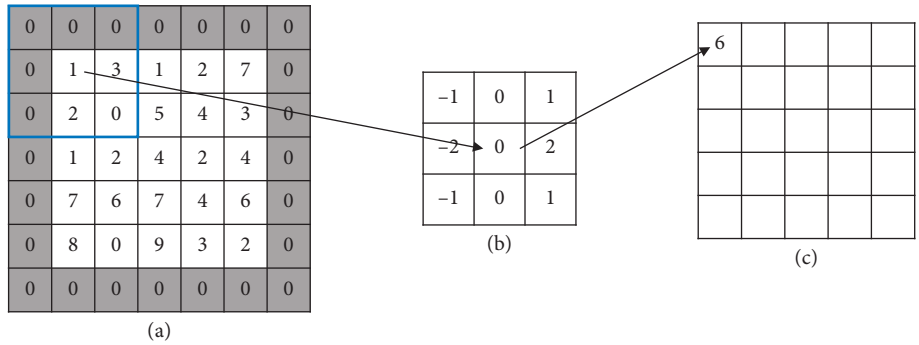


Figure 6: Convolution process.

Table 1: Parameters of autoencoder network structure.

Number of layers	Operation	Kernel size	Image size	Dimension
1	C	3×3	56×56	3
2	C	3×3	28×28	6
3	D	3×3	56×56	3
4	D	3×3	112×112	1

C denotes convolution and D denotes deconvolution.

The kernel size of the first layer of the network is 1×1. The number of network parameters can be greatly reduced by compressing the dimensionality of the encoder output from $28\times28\times6$ to $28\times28\times1$. Overfitting is prevented by using the dropout on the full connection of the fifth layer. The last layer contains three neurons. The activation functions of both the convolution and full connection are leaky ReLU functions, and a softmax function was used as the activation function for the last layer. The loss function is the cross-entropy,

$$c = -\frac{1}{n}\sum_{x}[y\ln a + (1-y)\ln(1-a)]. \quad (5)$$

(3) Addition of Fully Connected Network Layer. In a traditional convolutional self-encoder, the encoder and decoder network layers are convolution operations, while the convolution layer generally requires multiple convolution cores, which will cause the encoder output feature dimension to be too high. The decoder and encoder are connected by deconvolution, which makes it difficult to control the encoder output dimension. We improve the network structure of the self-encoder. After the convolution operation of the second layer of the encoder, we use the full connection layer to realize symmetry between the decoder and encoder. This solves the problem of the encoder output dimension control,

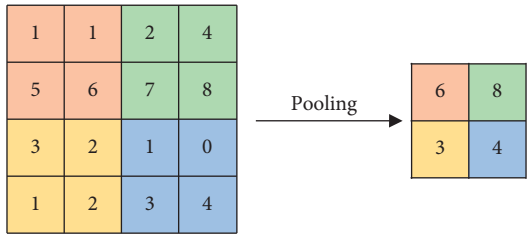


Figure 7: Pooling operation.

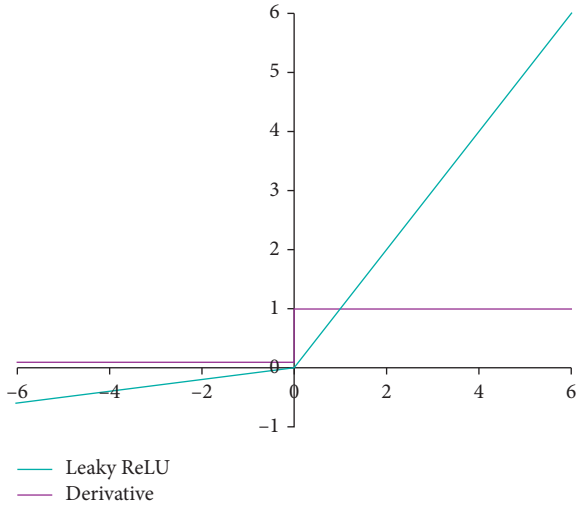


Figure 8: Leaky ReLU function and derivative.

Table 2: Parameters of convolutional fully connected network structure.

Network layer	Operation	Kernel size	Slip step	Number of neurons	Edge fill type
1	C	1×1	1×1	1	—
2	C	5×5	2×2	6	SAME
3	C	5×5	2×2	16	SAME
4	F	—	—	1024	—
5	F	—	—	3	—

C denotes the convolutional layer and F denotes the fully-connected layer.

it causes the convolution neurons to be connected, and the extracted features are more representative.

Figure 9 shows the network structure of the improved convolutional self-encoder. The network layer structure parameters are as follows:

(1) The first layer is an encoder input layer, and the input data dimension is (112, 112), with a single-channel gray image.

(2) The second layer network is a convolution layer with two convolution kernels of size (3, 3), the sliding step size is (2, 2), the padding operation is the same, the output dimension is a $(112/(2) \times (112/(2) = 56 \times 56$ characteristic graph, and the activation function is leaky ReLU.

(3) The third layer is a convolution layer with four convolution kernels of size (3, 3), the sliding step size is (2, 2), the padding operation is the same, the output dimension is $(56/(2) \times (56/(2) = 28 \times 28$, and the activation function is leaky ReLU.

(4) The fourth layer is a tile operation. The output of the previous network layer is tiled into one dimension, and the output dimension is 3136.

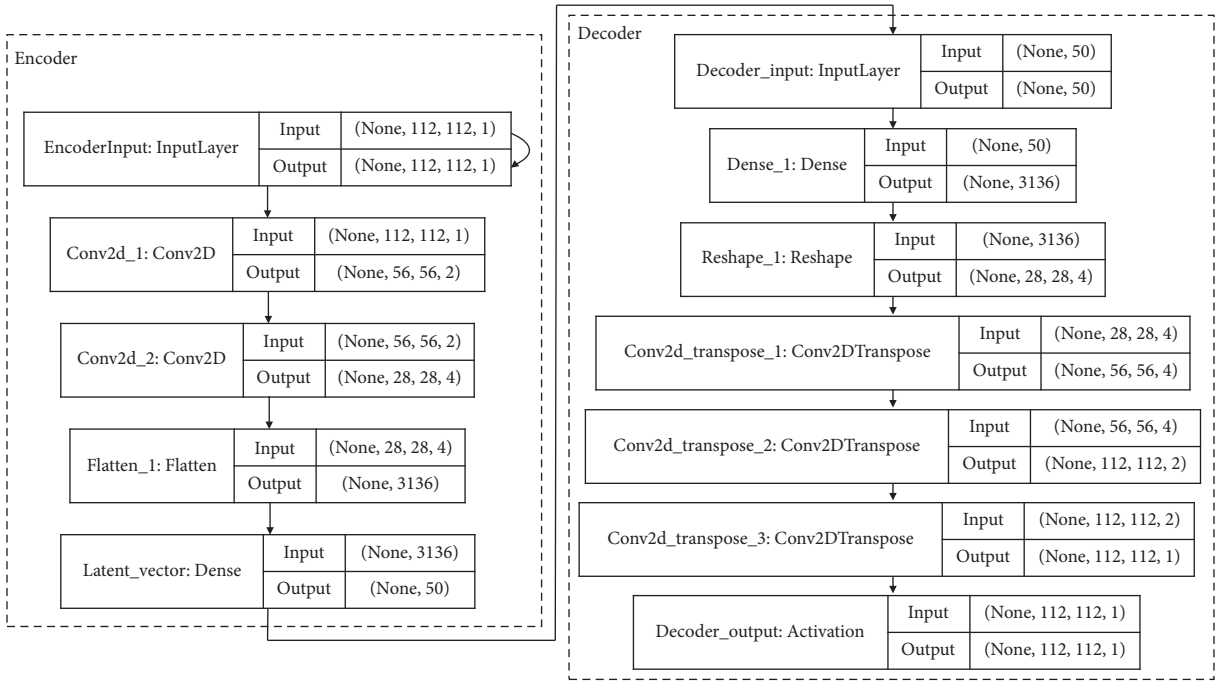


Figure 9: Network structure of self-encoder.

(5) The fifth layer is a full-connection operation with 50 neurons. The output is the encoder output, and the output dimension is 50. No activation function is used.

The network layer structure and parameters of the decoder are as follows:

(1) The first layer is a decoder input layer, with input dimension of 50.

(2) The number of neurons in the second layer is 3136, and the output dimension is 3136.

(3) The input data of the third layer has dimension 3136, and it is transformed to a feature graph of dimension (28, 28, 4).

(4) The fourth layer is deconvolution, the number of deconvolutions is 4, the convolution kernel size is (3, 3), the sliding step size is (2, 2), the output dimension is (56, 56, 4), and the activation function is leaky ReLU.

(5) The fifth layer is deconvolution, the number of neurons is 2, the size of the convolution nucleus is (3, 3), the sliding step size is (2, 2), the output dimension is (112, 112, 2), and the activation function is leaky ReLU.

(6) The sixth layer is deconvolution, the number of neurons is 1, the size of the convolution nucleus is (3, 3), the sliding step size is (1, 1), the output dimension is (112, 112, 1), and there is no activation function.

(7) The seventh layer is the activation layer. The sigmoid activation function is used to normalize the output value range of the decoder to (0, 1), and the output dimension is (112, 112, 1).

4. Experiment

The experimental environment was a 64 bit Windows 10 operating system, the processor was an AMD2600x, the graphics card was an Nvidia1060 6G, the memory was 16 GB, the programming language was *Python*, the backend of the deep learning framework was TensorFlow 1.13.2, and the interface language was Keras 2.3.1.

4.1. Training of Autoencoder. First, the autoencoder was trained. The weighting factors were initialized using a truncated normal distribution with a mean value of 0, a standard deviation of 0.1, a batch size of 64, and 1500 iterations. Figure 10 shows the reconstructed autoencoder. The first row contained the original defective samples, the second row contained the reconstructed samples of the autoencoder, and each column corresponded to a different defective sample. The results show that the autoencoder could successfully restore the overall structure and detailed local texture of the input samples, so the encoder efficiently extracted the features of the samples.

4.2. Training of Classification Network. The training and optimization of the classification network were performed using the Adam algorithm. The weighting parameters were initialized using a truncated normal distribution with a mean of 0, a standard deviation of 0.1, a bias of 0, a batch size of 64, and a dropout of 0.5. Figure 11 shows the changes in the training and test sets as the number of iterations increased. The accuracies of the test and training sets did not

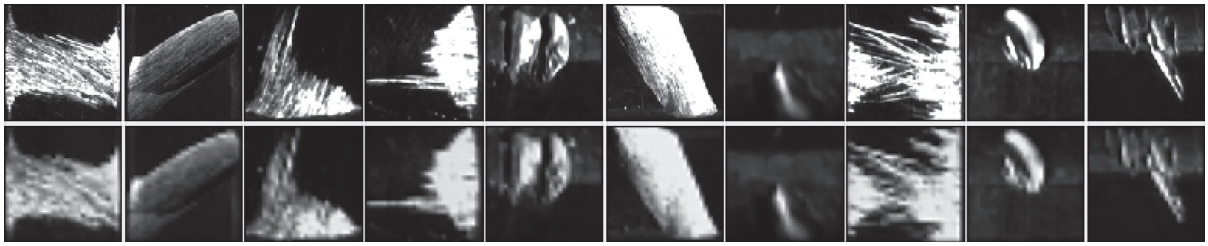


Figure 10: Reconstruction efficiency of the autoencoder.

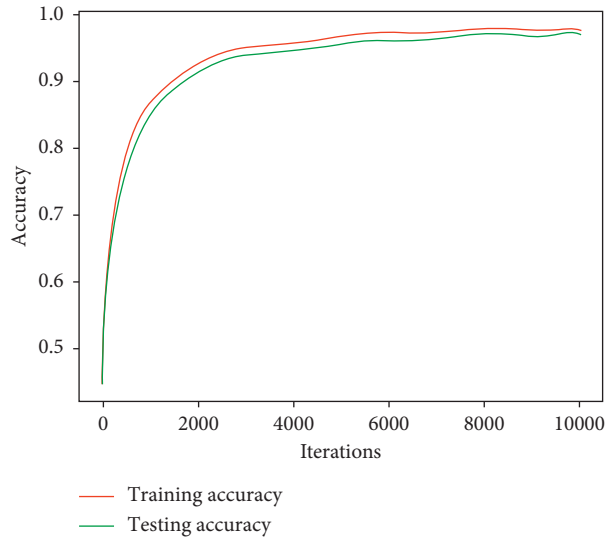


Figure 11: Relationship between accuracy and number of iterations.

Table 3: Comparison of accuracies (%) of various methods.

Classifier	Proposed network	Three-layer BP neural network	PCA + proposed network
Accuracy	98.13	95.24	91.23

Three-layer BP denotes backpropagation and PCA denotes principle component analysis.

differ significantly over the entire iteration process, and they remained within the acceptable range. In the first 1000 iterations, the accuracy increased rapidly, indicating that the optimizer found the correct direction for convergence at this time. The automatic increase of the learning rate by the optimizer facilitated rapid convergence. After 1000 iterations, the loss gradually approached the minimum value, and the optimizer switched to a slower learning rate to prevent the training from skipping over the optimal solution. The final accuracy of the model was 98.74% on the training set and 98.13% on the testing set.

4.3. Comparison of Experimental Results. To verify the performance of the network designed in this paper, the accuracies of the proposed network, a BP neural network, and classification using PCA-extracted features with the proposed classification network are compared in Table 3. The accuracy using features of the local defect area extracted using threshold segmentation and color conversion methods combined with the BP neural network was better than using PCA-extracted features with the proposed classification network but worse than that of the proposed algorithm alone. This showed that our autoencoder-based neural network classifier could indeed improve the classification accuracy of bearing defects.

5. Summary

To improve the classification accuracy of bearing surface defects, we designed a neural network classifier based on an autoencoder. Through experimental analysis and comparison with conventional algorithms, the algorithm proposed in this paper was shown not only to improve classification accuracy but to reduce the algorithm design workload of feature extraction, dimension reduction, and classification.

Compared to the PCA algorithm, we found that the features extracted by the improved autoencoder in this paper were more representative and could achieve higher classification accuracy.

Data Availability

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

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was funded by the National Social Science Fund of China (20BGL141).

References

[1] A. M. Al-Momani, M. A. Mahmoud, and M. S. Ahmad, "Factors that influence the acceptance of internet of things services by customers of telecommunication companies in Jordan," *Journal of Organizational and End User Computing*, vol. 30, no. 4, pp. 51–63, 2018.

[2] I. Kitouni, D. Benmerzoug, and F. Lezzar, "Smart agricultural enterprise system based on integration of internet of things and agent technology," *Journal of Organizational and End User Computing*, vol. 30, no. 4, pp. 64–82, 2018.

[3] K. G. Srinivasa, B. J. Sowmya, A. Shikhar, R. Utkarsha, and A. Singh, "Data analytics assisted internet of things towards building intelligent healthcare monitoring systems: iot for healthcare," *Journal of Organizational and End User Computing*, vol. 30, no. 4, pp. 83–103, 2018.

[4] S. K. Biswas, D. Devi, and M. Chakraborty, "A hybrid case based reasoning model for classification in internet of things (iot) environment," *Journal of Organizational and End User Computing*, vol. 30, no. 4, pp. 104–122, 2018.

[5] M. R. Reddy, K. G. Srinivasa, and B. E. Reddy, "Smart vehicular system based on the internet of things," *Journal of Organizational and End User Computing*, vol. 30, no. 3, pp. 45–62, 2018.

[6] M. Belkin and P. Niyogi, *Laplacian Eigenmaps for Dimensionality Reduction and Data Representation*, MIT Press, London, UK, 2003.

[7] P. Kunakornvong and P. Sooraksa, "Machine vision for defect detection on the air bearing surface," in *Proceedingsa of the 2016 International Symposium on Computer, Consumer and Control (IS3C)*, IEEE, London, UK, 2016.

[8] H. Çelik, L. C. Dülger, and M. Topalbekiroğlu, "Development of a machine vision system: real-time fabric defect detection and classification with neural networks," *The Textile Institute*, vol. 30, 2013.

[9] T. P. Minka, *Automatic Choice of Dimensionality for PCA*, MIT Media Lab, London, UK, 2000.

[10] M. Van and H. J. Kang, "Bearing defect classification based on individual wavelet local Fisher discriminant analysis with particle swarm optimization," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 1, pp. 124–135, 2017.

[11] J. Zapata, R. Vilar, and R. Ruiz, "Performance evaluation of an automatic inspection system of weld defects in radiographic images based on neuro-classifiers," *Expert Systems with Applications*, vol. 38, no. 7, pp. 8812–8824, 2011.

[12] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, no. 2, 2012.

[13] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *ICLR*, vol. 38, 2015.

[14] K. He, X. Zhang, S. Ren et al., "Deep residual learning for image recognition," 2015.

[15] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[16] D. E. Rummelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," *Nature*, vol. 323, no. 2, pp. 318–362, 1986.

[17] Y. Lecun, B. Boser, J. S. Denker et al., "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[18] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[19] K.-I. Funahashi, "On the approximate realization of continuous mappings by neural networks," *Neural Networks*, vol. 2, no. 3, pp. 183–192, 1989.

[20] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines vinod nair," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, Haifa, Israel, 2010.

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[22] C. Szegedy, V. Vincent, Sergey Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *IEEE Computer Society*, vol. 25, 2015.