Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 114 (2017) 341–348

Procedia Computer Science

www.elsevier.com/locate/procedia

Complex Adaptive Systems Conference with Theme: Engineering Cyber Physical Systems, CAS
October 30 – November 1, 2017, Chicago, Illinois, USA

The Application of One-Class Classifier Based on CNN in Image Defect Detection

Mei Zhang[a,*], Jinglan Wu[a], Huifeng Lin[a], Peng Yuan[a], Yanan Song[b]

[a]*South China University of Technology, School of Automation, Guangdong Guangzhou 510640, China P.R*
[b]*Guangdong University of Technology, College of Automation, Guangdong Guangzhou 510006, China P.R*

Abstract

In the field of defect detection, image processing algorithms and feature extraction algorithms have some limitations, owing to their necessity for extracting a large number of different features of diverse products images. Meanwhile, the images of defective products are less and various. Aiming at these problems, we presented a One-Class classifier based on deep convolution neural network to detect the defect images in this paper. We design a loss function with the penalty term based on Euclidean distance to train the deep convolution neural network model. A hypersphere is used as classification decision surface after setting an appropriate hypersphere radius according to the inspection accuracy. It maps the non-defective products into a hypersphere in a high dimensional feature space, while the defect images are mapped somewhere far from the center of hypersphere. Thus, a One-Class classifier based on convolutional neural network(CNN) model is proposed to detect the defects. Experiments show that the proposed method, with less number of iteration, help build the classifier for image defect detection with high generalization ability and high detection precision.

Peer-review under responsibility of the scientific committee of the Complex Adaptive Systems Conference with Theme: Engineering Cyber Physical Systems.

Keywords: One-Class classifier, CNN, Image defect detection, Hypersphere;

* Corresponding author. Tel.: +133-161-99180.
E-mail address: zhangmei@scut.edu.cn

1877-0509 © 2017 The Authors. Published by Elsevier B.V.
Peer-review under responsibility of the scientific committee of the Complex Adaptive Systems Conference with Theme: Engineering Cyber Physical Systems.
10.1016/j.procs.2017.09.040

1. Introduction

Defect detection usually refers to detecting the defect on the surface of objects. In recent years, the image defect detection has mainly relied on machine vision, which can detect the spots, pits, scratches, chromatic aberration and defect on the surface of work piece. It has been widely used to detect various products, such as textile, steel surface, metal, glass, paper and electronic components etc. With the increasingly stringent requirements for the quality of the products, the defect detection becomes an indispensable part in the industry.

The machine learning technology for defect detection mainly includes feature extraction and pattern classification. Aiming at diverse products images, lots of different feature extraction algorithms are proposed to extract a large number of different features, combining with diverse classifiers to solve the problems. Kumar[1] summarized the fabric defect detection methods and classified them into three categories: statistical, spectral and model-based. In statistical approach, Jiexian et al.[2] uses a gray level co-occurrence matrix to extract color and texture features , while a BP neural network is employed to classify the defects on flexible circuit surface. In spectral approach, Mak et al.[3] designed the Gabor filters to detect defects after extracting texture features using Gabor wavelet network. In model-based approach, Deng et al.[4] develops an anisotropic circular Gaussian Markovian Random Fields (ACGMRF) model to retrieve rotation-invariant texture features. However, the image defect detection is still faced with the problems of low contrast between the defect and non-defect area, the similarity of noise and fine defect, slow detection speed and low recognition accuracy. So more approaches are necessary for extracting the corresponding specific features.

The invariance properties of the above models are hand-designed in advance. In the method described in this paper however, the features do not come from prior knowledge about the same task, but learned from data automatically. To avoid the feature extraction procedure in the standard learning algorithms, without requiring specific information about the categories, we investigate an automated method for classifying the defect images from all images using a deep convolutional neural network. The CNN can automatically extract advanced features of the available image data with minimal pre-processing. As the first truly successful deep learning architecture due to the successful training of the hierarchical layers, CNN is proved to show outstanding ability in image classification. Miki et al.[5] employ the AlexNet network architecture to classify the tooth types on dental cone-beam computed tomography(CT) image without the need for precise tooth segmentation, with small number of training case, the paper gets the higher accuracy compared to the other algorithms. Soukup et al.[6] experimented with classical CNNs trained in pure supervised manner and explored the impact of regularization methods such as unsupervised layer-wise pre-training, which verified that CNN outperforms the model-based approach. Makantasis et al.[7] exploit a CNN to construct high level features and choose a Multi-Layer Perceptron as a detector, the detector achieves very fast predictions comparing against the other well-known supervised and semi-supervised learning approaches. Masci et al.[8] compared Max-pooling convolutional neural networks (MPCNNs) with SVM algorithm in the detection of seven defects in cold strips, showing the outstanding ability of feature extraction of CNNs.

The rest of the paper is organized as follows. Section 2 describes our problems, section 3 describes the considered deep neural network model as well as the one-class classifier that we employ. Section 4 presents the experimental protocol and the associated results, detailed performance study and analysis are also covered in this section. Finally, we discuss the future work and conclusion in Section 5.

2. The Problem Description

In this study, we aim at the defect detection problem with small and limited number of electronic component image dataset, as part of them are shown in Fig1. The non-defective samples are similar to each other, they can be rotated, shifted or flipped but should be complete like the first image in Fig.1. The defect images, however, are various such as the incomplete white part, incomplete gray part, the deformation, the spots and so on. We get a total number of 1090 images, where there are 600 non-defective images, and 490 defect images. Our goal in this paper is to sort out the defect images including those not present in the samples as much as possible with a limited number of images.

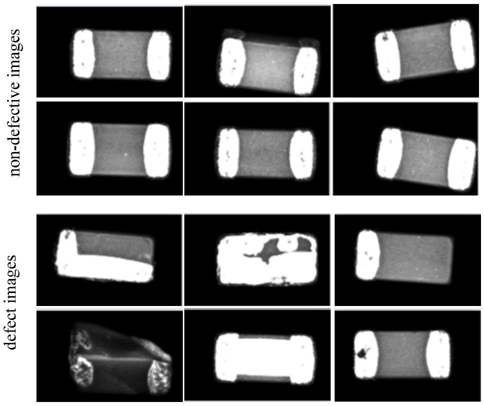


Fig.1. The image samples

Since the non-defective images are similar to each other, and we get the incomplete and limited variety of images, we focus on a one-class classifier based on CNNs to detect the defect on images in section 3. The image data are all manually labeled, among which 400 non-defective images and 300 defect images are used to train the network while the rest are employed to test.

3. Architectures

3.1. The CNN Model

Our investigation is based on a multilayer Convolutional Neural Network, AlexNet[9] . With its advantages of sparse interaction, parameter sharing and equivariant representation, CNN has already been successfully applied to image classification tasks, face detection, speech recognition and so on[10]. It is feasible to employ CNN to extract the high dimensional feature automatically from the image.

We employ a CNN architecture similar to AlexNet[11] but with different classifier and different training procedure. The network contains four layers with weights, the first three are convolutional and the remaining one is fully connected. We apply the technique proposed by Simonyan et al.[12] to use small stride of 3x3 kernel, and separate the layers by rectified linear unit (ReLU) activation, which is supposed to increase the nonlinearities inside the network and make the high dimensional feature of each image more discriminative. The output of the flatten layer is fed to 256 neurons, leading the original images of products to map to a 256-dimensional feature space. In order to reduce the load of the net, the input images are united as grayscale images and resized to 30×30 pixels. We take the defect image for example in Fig.2 to show CNN architecture for image defect detection and the output , which includes three convolutional, Relu and pooling layers and a dense layer. The number in the circle is the layer number, C(16,(3,3)) represents 16 convolution filters with 3x3 sizes. The dense layer maps the image into 256 dimensional feature space.

To demonstrate differences by colour difference and brightness and increase the amount of detail perceivable, we depict the feature maps in the form of heat map. As revealed in Fig.2, the network attempts to exact the main object in the first convolutional layers while the Relu layers add nonlinear factors to the network by constructing a sparse matrix that removes the redundancy in the data but retain characteristics in the iterative training. In the ninth layer , the network exact the sparse high-dimensional feature from the original image, inputting into a flatten layer. Compared the ninth layer in Fig.3(b) with that in Fig.3(a), there are more features in a defective image than in non-defective one. Such characteristics is also reflected in the comparison between Fig.2 and Fig.3(a), corresponding to the greater distance between the defect image and non-defective one. Owing that CNNs are trained in supervised manner, it can adjust the weights of every convolution filter according to the loss function to exact the special features for defect images to increase the distance to the non-defective, but exact the similar features between non-defective images.

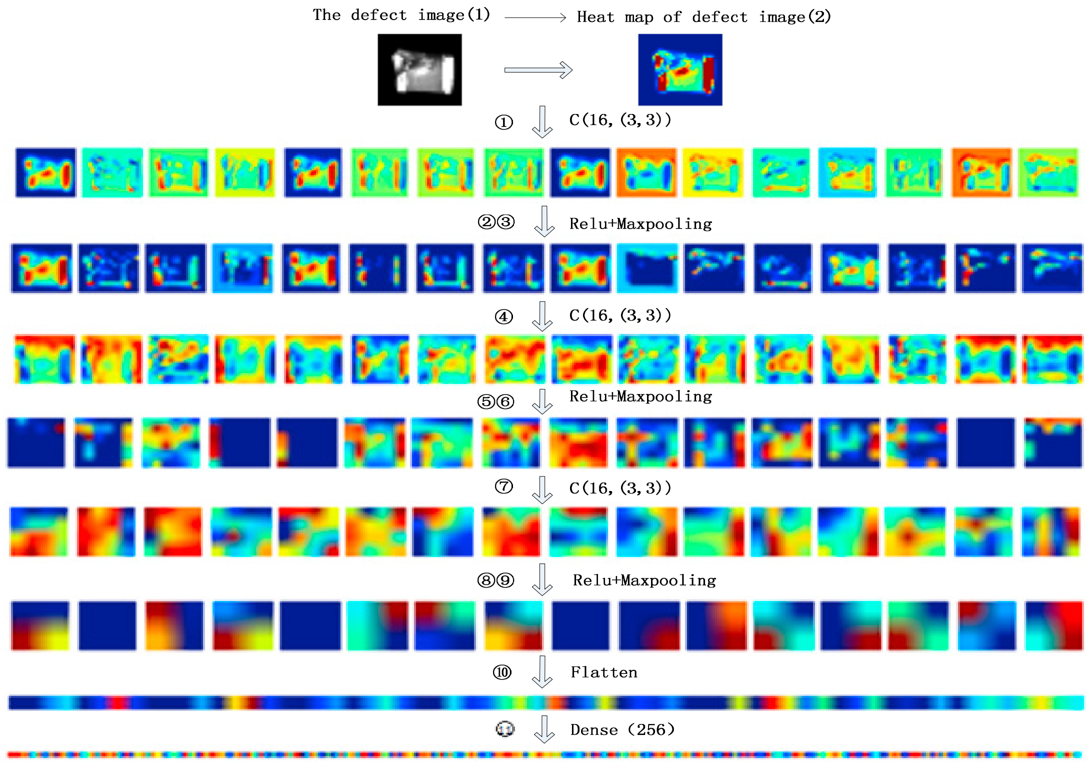
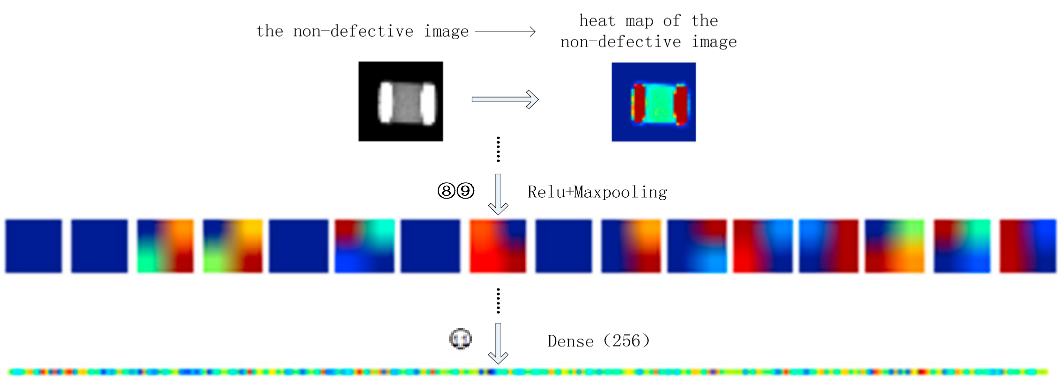
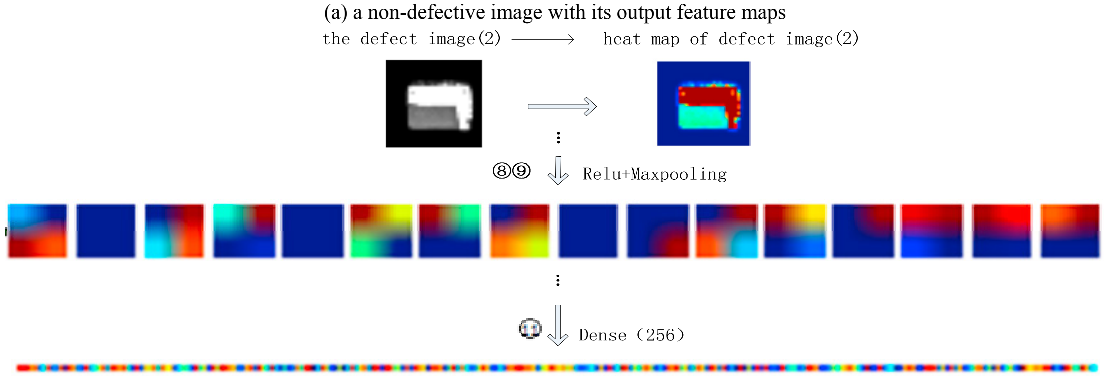


Fig.2. The CNN architecture for image defect detection



(a) a non-defective image with its output feature maps



(b) a defective image with its output feature maps

Fig.3. The comparison between a non-defective image and defect one.

3.2. The One-Class Classifier

As expressed by Chen et al.[13], lots of problems can be regarded as a strict two-class classification problem, with equal treatments on both positive and negative examples, the same as the defect detection problems. But it is reasonable to assume that the non-defective examples cluster in certain way while the defect examples usually do not cluster since there are various defects and they can belong to any class. Since the non-defective images are similar to each other, and we get the limited variety defect images, we consider the defect detection problem as a One-Class problem by designing a One-Class classifier as depicted in Fig.4.

Our strategy is to use the CNNs to map the non-detective samples into a high-dimensional hypersphere, represented as a dashed circle in Fig.4 to describe the data in high-dimensional feature space, and then try to put most of the non-defective features into the hypersphere, while the defect features are far away from the hypersphere. The radius, however, is setting according to the inspection accuracy, a large radius is allowed if the rough defect detection is permitted, a small radius can divide images with slight defect out. Therefore, when a image is mapped outside the hypersphere, we classify it as a defect image.

We assume that non-defective samples cluster in a hypersphere in a certain feature space, while defect samples are outside the hypersphere. Our goal is to find out the mapping function between the raw samples and their representations in this feature space, and find out the center of the hypersphere. In training phase, when given a batch of training samples, we first calculate the cluster center in the feature space of the non-defective samples. We choose the mean of the outputs corresponding to the non-defective samples as the cluster center. We suppose $(\mathrm{X}_i, \mathrm{Y}_i)$ be the i-th labeled sample of the batch, where Y_i is a binary label attached to X_i. We define $Y_i = 1$ if X_i is a non-defective sample, while $Y_i = 0$ if X_i is a defect sample. Thus, we have equation(1):

$$center = \sum_{i=1}^{n} [(1 - Y_i) F_W(X_i) + Y F_W(X_i)] / \sum_{i=1}^{n} Y_i \tag{1}$$

where $F_W($ is the output of in the training session. Then we calculate the Euclidean distance between each output and the cluster center by equation(2) :

$$D_W(\mathrm{X}) = \| F_W(\mathrm{X}) - center \|_2 \tag{2}$$

The loss function we use is the contrastive loss function refer to the paper[14] by equation(3) :

$$L(\mathrm{W}, \mathrm{Y}, \mathrm{X}) = \frac{1}{2} \mathrm{Y} \mathrm{D}(\mathrm{X})^2 + \frac{1}{2} (1 - \mathrm{Y}) \{\max(0, \mathrm{m} - \mathrm{D}_W)\}^2 \tag{3}$$

So the loss function of a whole batch can be shown as equation(4):

$$L(\mathrm{W}) = \sum_{i=1}^{n} L(\mathrm{X}_i, \mathrm{Y}_i) \tag{4}$$

Note that the cluster center is not fixed, it changes adaptively while training according to equation(1). Fig.4 is the illustration of the One-Class classifier in the course of clusting, where the dashed circle represents the hypersphere, and the white dot is the cluster center of hypersphere while the black dot is the final cluster center of non-defective samples after Iterative training. The red dots are the high dimensional features of different defect images while the blue dots are the high dimensional features of different non-defective images. In the process of training, the non-defective samples approach the center point gradually and the center is constantly updated, meanwhile, the defect samples are given penalty if it is mapping inside the hypersphere, which is depicted as the arrow tip in Fig.4. After training, we obtain the mapping function $F_W(\mathrm{X})$ and the center of the hypersphere, which we can use to classify new samples. When given a new sample, we first calculate its representation in the feature space using $F_W(\mathrm{X})$. Then we calculate the Euclidean distance between the feature vector and the center of the hypersphere. If the distance exceeds a threshold value which is set up manually depending on the needed accuracy, it would be classified as a detective one, otherwise, it would be considered as a non-defective one. Defective samples are still necessary in the training process of our One-Class classifier, but we still call it One-Class classifier because its performance does not largely depend on the number and variety of defective training samples when compared to a Two-Class classifier.

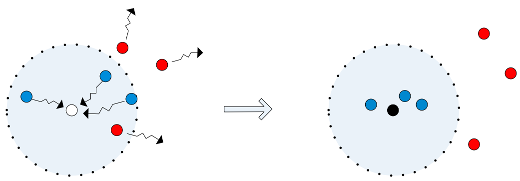


Fig.4. The illustration of the One-Class classifier in the course of clusting.

4. Experimental and results

We evaluate the CNN with One-Class classifier by comparing it to a typical Two-Class classifier CNN model with softmax classifier. Both models are almost the same in the architecture in Fig.2 except that the Two-Class classifier CNN model has an additional two-neurons output layer fully connected to the last layer of the One-Class classifier. The loss function of the One-Class classifier is the contrastive loss function we describe above, while the Two-Class classifier is the softmax loss. We employ the Adam(Adaptive Moment Estimation) as the optimizer and evaluate the result by ROC (receiver operating characteristic curve) and AUC (Area Under Curve).

We first experiment with 400 non-defective images and 300 defect images to train the network while the rest are employed to test. The result shows in Fig.5 where CNN with one class classifier performs well with AUC equal to 0.9889 under the case that AUC equal to 1 would be perfect.

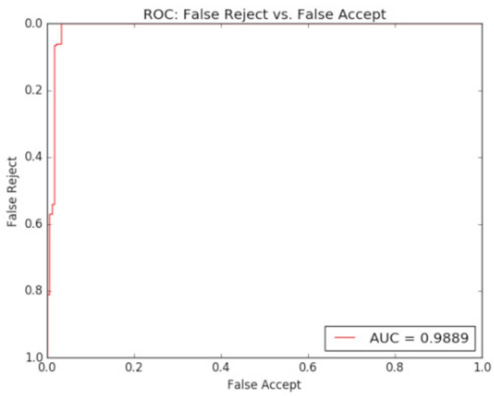


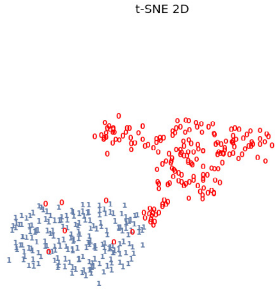
Fig.5. The ROC curve of the CNN with one-class classifier

After a threshold is given, we can obtain the training samples' classification accuracy according to the confusion matrix of training samples. Therefore we can select the threshold value corresponding to the highest accuracy of training samples' classification, to classify the test samples. In our experiments, we select the threshold as 0.35, that is the radius of the hypersphere is 0.35, then the confusion matrix is shown in Table 1, where all the non-defective images are classify correctly and 6 defect images are classified incorrectly. The images classified incorrectly are shown in Fig.6(a) due to that they have subtle flaws, making them similar to non-defective images.

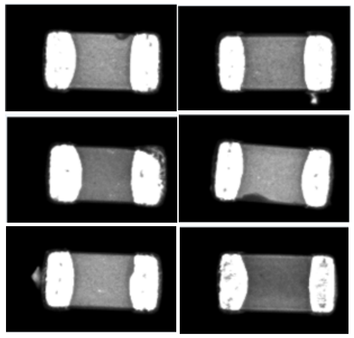
Table 1. The confuse matrix of the CNN with one-class classifier.

	Predict to be defect	Predict to be non-defective	Total
Defect images	184	6	190
Non-defective images	0	200	200
Total	184	206	390

According to the principle that data points similar in high dimensional space are also similar when mapped to low dimensional space, we visualize the high dimensional data by t-SNE[15] as shown in Fig.6(a), where non-defective samples cluster within a certain space and the defect samples scatter somewhere. The CNN with One-Class classifier works consistent with our intention to classify out defect samples as much as possible.



(a)

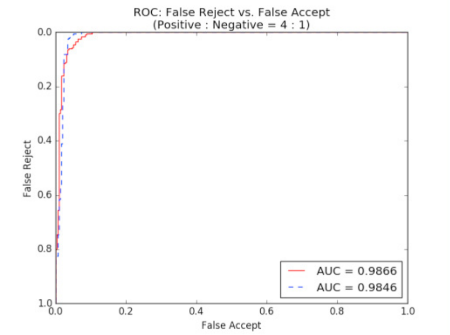


(b)

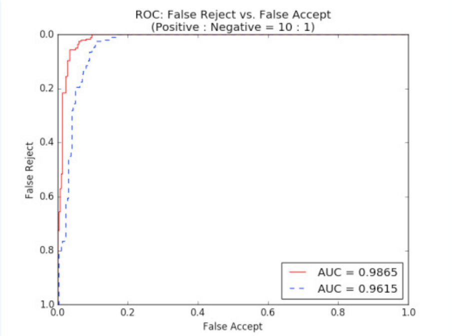
Fig.6. Visualization of classification results. (a)Visualization of the samples. The blue ones are the non-defective samples with label one, while the red zeros are the defect samples with label zero. (b): The defect images that is classified incorrectly to be non-defective.

To explore the potential robustness of the classifier and simulate the case of varying degrees of scarcity of defective samples, we select four ratios between non-defective samples and defective samples in the training set as 4:1, 10:1, 20:1 and 40:1 only by reducing the number of defective samples to train our model and the basic CNN. The test data for the four experiments are designed to be the same and fixed. The contrast of the two models is depicted in Fig.7.

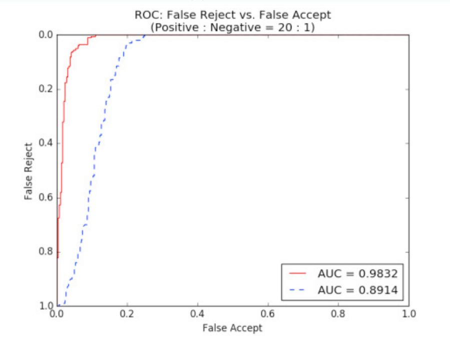
As shown in Fig.7, with the reduction of defect samples, our model has always maintained a good performance, relatively, the performance of basic CNN with softmax classifier is getting worse. We conclude that the model is suitable for the case with small number of defective images, and the CNN with one-class classifier is robust.



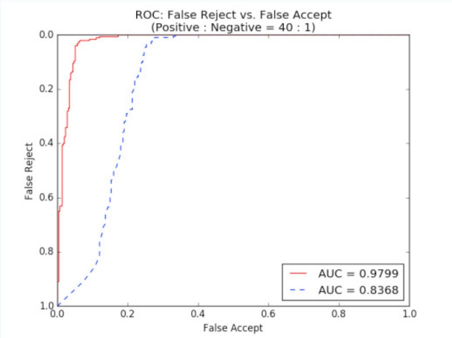
(a) 4:1



(b) 10:1



(c) 20:1



(d) 40:1

Fig.7. The real line is the CNN with one-class classifier and the dotted line is CNN with softmax. “Positive” denotes non-defective samples while “Negative” denotes defective samples.

5. Conclusion and future work

In this paper, we develop a CNN with One-Class classifier to deal with the defect detection problem with small limited samples. A user can appropriately select the threshold value depending on the requirements for detection accuracy. Experiments show that the One-Class classifier performs better than the Two-Class classifier, and the One-Class classifier is more robust. Although the results of the proposed scheme have been demonstrated for the electronic component, the scheme can potentially be used for other products.

Some more research should be done in how to choose appropriate threshold value for the hypersphere. Also, the algorithm requires defect samples during training, which is the limitation of the algorithm.

Acknowledgements

This research is supported by China Scholarship Council (CSC) program (Grant No. 201706155094) and Science and technology project of Guangdong Province of China (Grant No. 2014A020217015 and 2016A020222012).

References

[1] Kumar A. Computer-vision-based fabric defect detection: A survey[J]. IEEE transactions on industrial electronics, 2008, 55(1): 348-363.

[2] Jiexian H, Di L, Feng Y. Detection of surface of solder on flexible circuit[J]. Optics and Precision Engineering, 2010, 18(11): 2443-2453.

[3] Mak K L, Peng P. An automated inspection system for textile fabrics based on Gabor filters[J]. Robotics and Computer-Integrated Manufacturing, 2008, 24(3): 359-369.

[4] Deng H, Clausi D A. Gaussian MRF rotation-invariant features for image classification[J]. IEEE transactions on pattern analysis and machine intelligence, 2004, 26(7): 951-955.

[5] Miki Y, Muramatsu C, Hayashi T, et al. Classification of teeth in cone-beam CT using deep convolutional neural network[J]. Computers in Biology and Medicine, 2017, 80: 24-29.

[6] Soukup D, Huber-Mörk R. Convolutional neural networks for steel surface defect detection from photometric stereo images[C]. International Symposium on Visual Computing. Springer International Publishing, 2014: 668-677.

[7] Makantasis K, Protopapadakis E, Doulamis A, et al. Deep convolutional neural networks for efficient vision based tunnel inspection[C]. Intelligent Computer Communication and Processing (ICCP), 2015 IEEE International Conference on. IEEE, 2015: 335-342.

[8] Masci J, Meier U, Ciresan D, et al. Steel defect classification with max-pooling convolutional neural networks[C]. Neural Networks (IJCNN), The 2012 International Joint Conference on. IEEE, 2012: 1-6.

[9] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[C]. Advances in neural information processing systems. 2012: 1097-1105.

[10] Liu W, Wang Z, Liu X, et al. A survey of deep neural network architectures and their applications[J]. Neurocomputing, 2017, 234: 11-26.

[11] Bromley J, Bentz J W, Bottou L, et al. Signature Verification Using A "Siamese" Time Delay Neural Network[J]. IJPRAI, 1993, 7(4): 669-688.

[12] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.

[13] Chen Y, Zhou X S, Huang T S. One-class SVM for learning in image retrieval[C]. Image Processing, 2001. Proceedings. 2001 International Conference on. IEEE, 2001, 1: 34-37.

[14] Hadsell R, Chopra S, LeCun Y. Dimensionality reduction by learning an invariant mapping[C]. Computer vision and pattern recognition, 2006 IEEE computer society conference on. IEEE, 2006, 2: 1735-1742.

[15] Maaten L, Hinton G. Visualizing data using t-SNE[J]. Journal of Machine Learning Research, 2008, 9(Nov): 2579-2605.