



Using convolutional neural networks for character verification on integrated circuit components of printed circuit boards

Chun-Hui Lin¹ · Shyh-Hau Wang^{1,2} · Cheng-Jian Lin³

Published online: 21 May 2019

© Springer Science+Business Media, LLC, part of Springer Nature 2019

Abstract

Manufacturers of printed circuit boards (PCBs) typically use automated optical inspection (AOI) machines to test their PCBs. However, AOI machines employ conventional image-processing methods. If the integrated circuit (IC) components are not identical to the golden samples, then the AOI machine registers those IC components as flaws. Conventional image-processing methods cause misjudgments and increase the cost of manual reviews. Character-verification and image-classification systems are proposed in this paper for detecting misplaced, missing, and reversed-polarity parts. The regions of IC components can be identified on PCBs by using the contour border-detection method. Through the proposed convolutional neural network (CNN) structure and refinement mechanism, the characters can be successfully recognized. The image-classification system was applied only to images with blurry characters. Different CNN learning structures were used in both systems, and the refinement mechanism was used in both systems to improve the results. The proposed character-verification and image-classification methods achieved 98.84% and 99.48% passing rates, and the amount of required training time was less than that of other methods, demonstrating the proposed methods' greater effectiveness.

Keywords Printed circuit board · Convolutional neural networks · Component testing · Contour detection · Deep learning

1 Introduction

With developments in computer technology, communication, and ecommerce, printed circuit board (PCB) requirements have become increasingly strict. Additionally, surface-mount technology has exhibited steady progress. To increase the quality and stability of PCBs, some problems must be addressed, such as the misplacement, loss, and reversed polarity of PCB parts.

Automated optical inspection (AOI) machines are commonly used by manufacturers. Workers must spend time establishing each parameter, including light colors, light angles, image lighting, and the image contrast ratios, prior to testing. Pattern matching is then used to compare the testing images

with the golden samples. Unstable image quality may result in workers being required to retest the images, thus increasing companies' labor costs.

These problems can be solved using two methods: the match method and character verification method. Match methods, such as that developed by Cho et al. [1], use a pattern-matching algorithm to compare the input images with the standard images to identify the misplaced parts of the PCB. The images are then converted through discrete wavelet transformations to boost accuracy and stability. This pattern-matching method was employed by Crispin et al. [2]. However, they used a genetic algorithm to expedite the identification process of the PCB. The pattern-matching method demonstrates excellent performance and accuracy but is easily affected by image lighting, component deviation, and noise.

The primary aim of character verification is to verify laser or ink jet words on integrated circuit (IC) components while checking for missing and misplaced parts. Lee et al. [3] proposed feature extraction through Gabor filter composition, direction gradient, wavelet coefficient, and difference in edge spacing. After feature extraction, the AdaBoost is used for classification. The stroke width transform was proposed by Epshtein et al. [4]. The contour border-detection algorithm and direction gradient are also used in this method. After

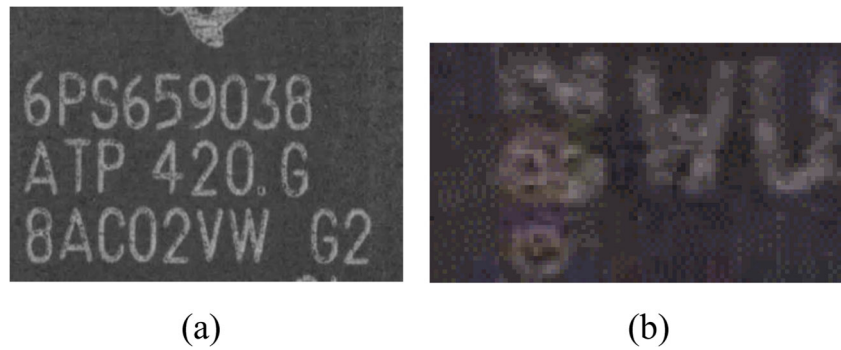
✉ Cheng-Jian Lin
cjlin@ncut.edu.tw

¹ Department of Computer Science & Information Engineering, National Cheng Kung University, Tainan 701, Taiwan

² Intelligent Manufacturing Research Center, National Cheng Kung University, Tainan 701, Taiwan

³ Department of Computer Science & Information Engineering, National Chin-Yi University of Technology, Taichung 411, Taiwan

Fig. 1 (a) Clear-character IC component (b) Blurry-character IC component



defining the stroke width, the connection method and predefined threshold are used to divide the character areas.

For character verification, Nava et al. [5] extracted the features through PCA and then classified the characters by using the conditional probability of the Bayesian function. Neullens et al. [6] evaluated the performance of various methods of optical character recognition (OCR) and proposed preprocessing steps for improving performance and stability.

Deep neural networks (DNNs) are considered a basic tool for extracting features from training data, and studies on character detection and verification have increasingly used DNN. For example, to solve the problem of small characters, which are difficult to detect, the feature enhancement network was proposed by Zhang et al. [7]. Furthermore, Zhang et al. [7] designed the adaptive position-sensitive region-of-interest pooling layer to improve accuracy. Shi et al. [8] implemented a character-verification system in an end-to-end network. First, features were extracted using a convolutional neural network (CNN). Next, the map-to-sequence method was used to transform these features into feature vectors. Finally, a recurrent neural network (RNN) and connectionist temporal classification method were used to verify the words.

The object-detection method using deep learning with graphic processing unit hardware [9, 10] exhibits adequate performance in many tasks; however, the considerable time required for training renders this detection method inefficient.

The major contributions of this study are described as follows:

- A character-verification system with deep learning is proposed to recognize IC components in images with clear characters.
- An image-classification system is also proposed to classify the images without characters or those with blurry characters by CNN structure.
- A novel refinement mechanism is used in both systems. It refines the CNN output score and increases the accuracy for detecting misplaced, missing, and reversed-polarity parts.
- The experiments are implemented in both systems in this study. The experimental results indicate that the proposed method exhibited a superior passing rate and less training time compared to the other methods.

The remainder of the paper is organized as follows. Section 2 introduces the proposed methods of this study. Details on the character-verification method and the image-classification method are described in Sections 3 and 4. Section 5 presents the experimental results, and Section 6 offers conclusions for this study.

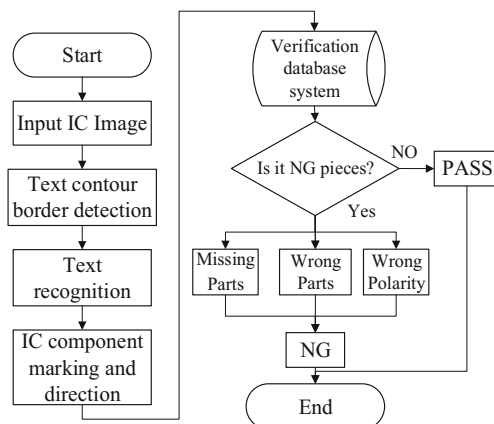


Fig. 2 Structure of the character-verification system

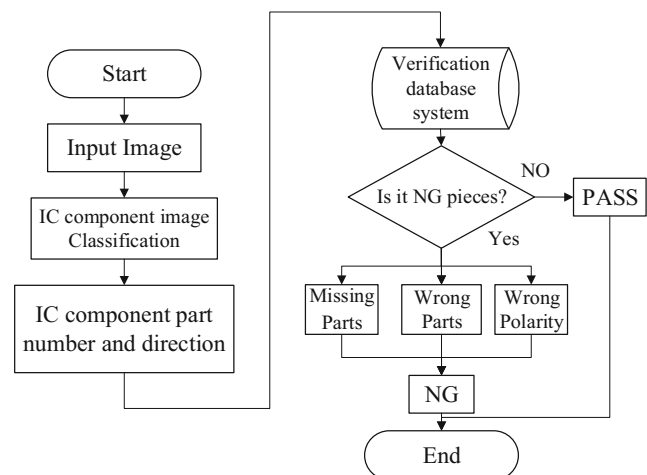


Fig. 3 Structure of the image-classification system



Fig. 4 Contour detection

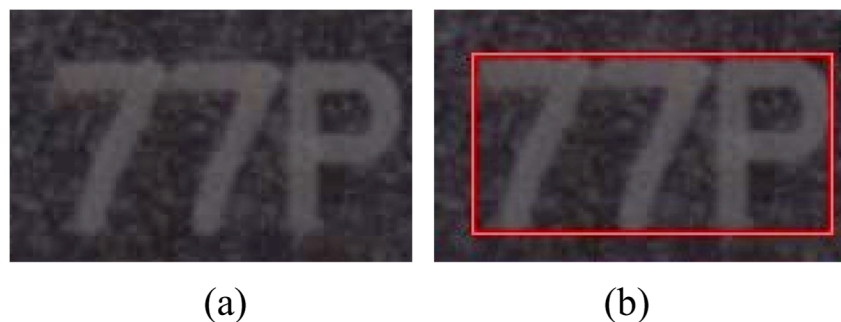
Table 1 CNN structure

| Input | Size 1*28*28 |
|------------------|---|
| Convolution1 | Kernel Size: 5, Pad:2, Stride:1, Output:64 |
| Activation1 | Type: ReLU |
| Pooling1 | Kernel Size: 2, Stride:2, Type: MAX |
| Convolution2 | Kernel Size: 5, Pad:2, Stride:1, Output:128 |
| Activation2 | Type: ReLU |
| Pooling2 | Kernel Size: 5, Stride:2, Type: MAX |
| Fully connected3 | Output:300, Dropout:0.5 |
| Activation3 | Type: ReLU |
| Fully connected4 | Output:120, Softmax |

2 Proposed methods

The images requiring testing were captured from the PCB by using high-magnification camera lenses through the AOI machine, and then the region of interest was analyzed to determine the IC position. The images might therefore contain some noise, such as uneven light, skewed angles, or a low degree of contrast. Two types of images were examined: those with clear characters, such as in Fig. 1(a), and those with blurry characters, as in Fig. 1(b).

Fig. 5 (a) Original image (b) After contour detection



Two systems, a character-verification system and an image-classification system, are proposed in this study. The primary goals of these systems are to achieve high accuracy and performance and decrease the number of manual-adjustment parameters. Figure 2 displays the structure of the character-verification system that is used to examine the clear-character IC component indicated in Fig. 1(a).

Figure 3 Structure of the image-classification system that is used in the blurry-character IC component presented in Fig. 1(b).

3 Deep learning for character verification

3.1 Contour border detection

Contour border detection is the preprocessing of the character-verification system. First, the images were converted into grayscale after being input into the system. Next, Gaussian smoothing was used to remove noise from the images, and Otsu [11] was used to automatically reduce noises and change the grayscale images into binary images. The border-following algorithm developed by Suzuki [12] was used to extract the characters that did not belong in the background. The border-following algorithm is used to derive chain codes from the border between a connected 1-pixel component and 0-pixel (background) component. The 8-connectivity is defined to search the white border but not the inside of the character. Figure 4 reveals that the detected rectangle frame was the area of the character.

3.2 CNN structure of character verification

The LeNet-5, consists of two sets of convolutional and average pooling layers, and two fully-connected layers, is a straightforward and well-known architecture for character recognition. In this study, the CNN structure which is modified from LeNet is provided in Table 1. To meet the manufacturer requirements, including those for training speed and the accuracy, the sizes of the grayscale images were set to $1 \times 28 \times 28$. The 64 and 128 5×5 convolution kernels were the best parameter set which were used in the first and second

Fig. 6 (a) Original image (b) After contour detection

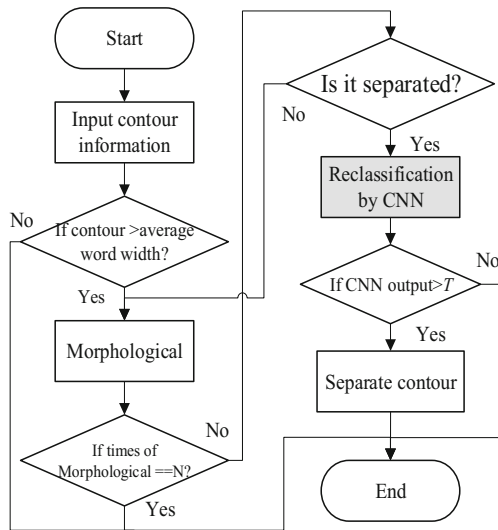
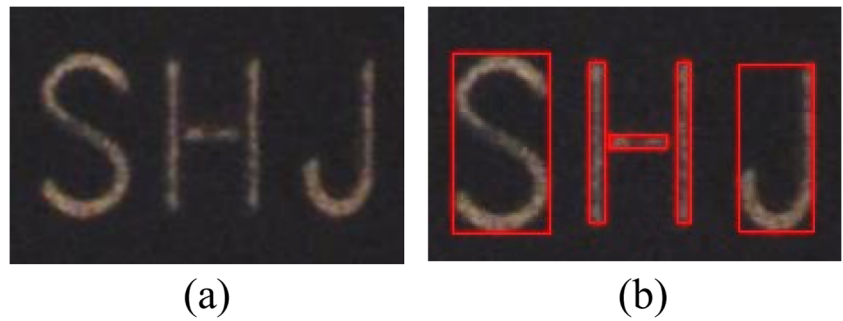


Fig. 7 Procedure for character connection

convolution layers to extract the features after training and testing experiments. The padding was required to remain the same size after convolution. The rectified linear unit (ReLU) activation function was used to strengthen the feature expression, and pooling was used to reduce the size of the feature images. Flattening was employed to connect the feature images to the fully connected layer, and the dropout was set to 0.5 during training; abandoning 50% of the neurons could prevent overfitting.

Equation (1) demonstrates the use of the probability distribution P of c types through softmax to acquire the maximum

probability type [22], the final result Y , as presented in Eq. (2).

$$p_i = \frac{e^{z_i}}{\sum_{j=1}^c e^{z_j}} \tag{1}$$

$$Y = \arg_{c \in [1, C]} \max(p(c|X)) \tag{2}$$

where p_i is the result of softmax, z_j is the output before softmax, c is the number of types, and e is the exponential.

3.3 Refinement mechanism of character verification

After being trained by the CNN structure, the characters, numbers, IC logos, symbols, and angles can be successfully and roughly recognized. However, the image quality might sporadically become unstable. Contour detection can solve problems associated with the system being unable to recognize a character. Perceiving multiple characters as one character is one such problem. Figure 5 provides an example of three characters being detected as one character because of noise or a lack of light. Another problem is blurry or fractured characters. Light and low print quality cause the characters to be separated into multiple regions, as demonstrated in Fig. 6.

The refinement mechanism for contour detection is introduced as follows, with three cases presented for solving the two aforementioned problems.

- Character connection

To prevent the system from recognizing multiple characters as one, the opening operation in mathematical morphology was used. The procedure is displayed in Fig. 7.

Fig. 8 (a) Vertical fracture (b) Horizontal fracture

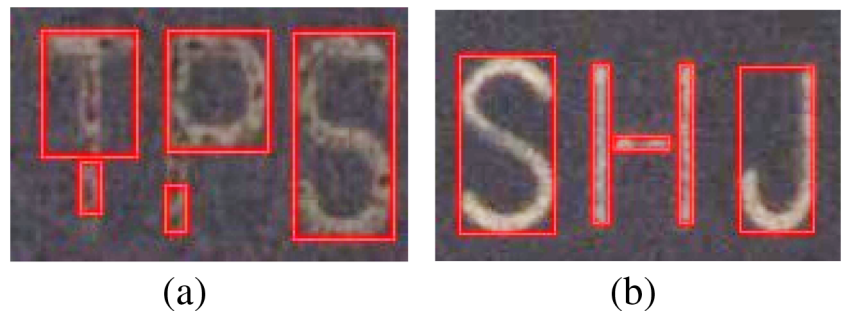
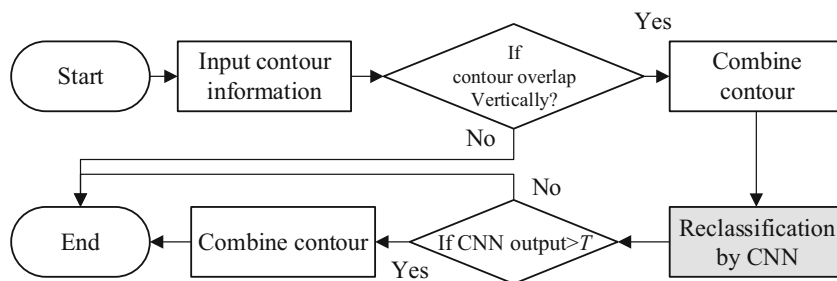


Fig. 9 Procedure of vertical character fracture



First, the areas of all contour regions, referred to as the word width, were calculated. Next, for the opening operation to the contour region, words with a width greater than the average word width were identified. The opening operation was then tested N times until the system split the contour region. The number N was set to 5 in the experiments. Finally, the region was recognized again by the CNN. An output score greater than the threshold T indicated that the character had been accurately divided; otherwise, the word had been inaccurately split.

The two directions of fractured characters are illustrated in Fig. 8.

- Vertical character fracture

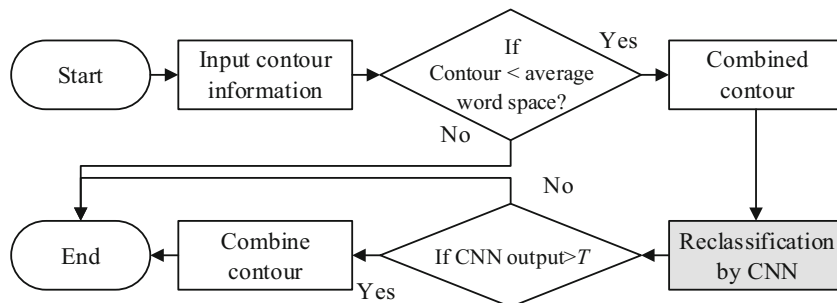
The mechanism of vertical character fracture is presented in Fig. 9. In the contour region, the string projects downward, and if any overlap occurs, then they are combined and then again recognized by the CNN. An output score greater than the threshold T indicates that the character was successfully connected.

- Horizontal character fracture

The flowchart of horizontal character fracture is displayed in Fig. 10. First, the angle of the string is determined. Next, the distance (word space) of each contour is calculated and the contour regions that are smaller than the average word space are combined. Finally, the region is recognized again by the CNN. An output score greater than the threshold T indicates that the character was successfully connected.

In this study, the proposed adaptive threshold T was set by using training data. The lowest score of all the output scores from the character-verification model was set as threshold T.

Fig. 10 Procedure of horizontal character fracture



In order to compare with adaptive threshold T, the real scores from CNN are needed therefore the softmax which regularizes the output scores from 0 to 1 was removed in the testing phase [22]. Eq. 3 demonstrates the output type without softmax.

$$z_i = pooling(ReLU(W \otimes X + b)), i \in [1, C] \tag{3}$$

where X is the input image, W is the weight after training, b is the bias, \otimes is the convolution operation, $ReLU(\cdot)$ is the non-linear activation function, $pooling(\cdot)$ is the pooling operation, and z_i is C types of outputs without softmax.

4 Deep learning for image classification

Figure 1 (b) displays the blurry-character or no-character IC component images used in this mechanism. The proposed image-classification method involves using the CNN structure to detect images and to determine the direction of the IC component and the item number that the IC component belongs to.

4.1 CNN structure of image classification

The CNN structure of image classification was revised from AlexNet. A champion of ImageNet Large Scale Visual Recognition Challenge in 2012 with five convolution layers and three fully-connected layers, however, in this study, less layers than AlexNet were used. Only four feature-expression layers were designed in the structure, and then three fully connected layers were connected. To prevent overfitting, the dropout was set at 0.3 and 0.5 for the first two fully connected

Table 2 Structure of IC component image classification

| Input | Size 3*250*250 |
|------------------|---|
| Convolution1 | Kernel Size: 7, Pad:3, Stride:1, Output:48 |
| Activation1 | Type: ReLU |
| Pooling1 | Kernel Size: 2, Stride:2, Type: MAX |
| Convolution2 | Kernel Size: 5, Pad:2, Stride:1, Output:96 |
| Activation2 | Type: ReLU |
| Pooling2 | Kernel Size: 3, Stride:2, Type: MAX |
| Convolution3 | Kernel Size: 5, Pad:2, Stride:1, Output:96 |
| Activation3 | Type: ReLU |
| Pooling3 | Kernel Size: 5, Stride:2, Type: MAX |
| Convolution4 | Kernel Size: 3, Pad:2, Stride:1, Output:128 |
| Activation4 | Type: ReLU |
| Pooling4 | Kernel Size: 5, Stride:2, Type: MAX |
| Fully connected3 | Output:250, Dropout:0.5 |
| Activation3 | Type: ReLU |
| Fully connected3 | Output:150, Dropout:0.5 |
| Activation3 | Type: ReLU |
| Fully connected4 | Output:24, Softmax |

layers in the training phase. Six classes were then output from a probability distribution using softmax. The structure of image-classification is presented in Table 2. In this study, we try to increase the number of layer. The results of image classification cannot be significantly improved and requires a longer training time.

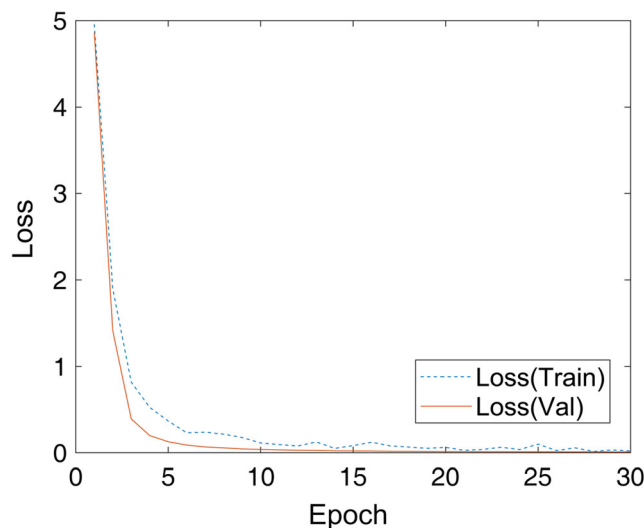
4.2 Refinement mechanism of image classification

After training, the IC component images were input in CNN structure and the Top-1 predicted answer was selected as its class; the IC part number and its angle could then be determined. Consequently, misplaced parts and polarities could be identified by the system. However, if the probability distribution of the output closely corresponded to all classes, then the image class was difficult for the CNN to predict.

To address the above-mentioned problem, in the testing phase, the softmax was removed and the system recorded all of the scores from the training data of different image classes. The lowest score was set as the threshold of its class. Therefore, the input image achieving a score that is lower than the threshold was considered abnormal (misplaced or missing parts).

Table 3 Training parameters of character verification

| Training Epochs | Batch Size | Optimizer Type | Base Learning Rate |
|-----------------|------------|----------------|--------------------|
| 30 | 300 | Adam [13] | 0.0001 |

**Fig. 11** Learning graph of character verification

5 Experimental results

To verify the efficiency of the proposed method, real IC components were used in the experiments. The system was evaluated to determine whether it could identify all of the strings and angles from the IC components, and a standard pass rate was used. If characters were not appropriately compared with the database, then they were classified as a misplaced component. If no character was detected, then it was regarded as a missing part. If the system verified the characters correctly but the angle was incorrect, then it was classified as being the wrong polarity.

For image classification, the system was evaluated to determine whether it could display the number and the right angle of the IC component, and a standard passing rate was used. Testing was conducted using the deep-learning tools Caffe and NVIDIA Digits, as well as using a learning environment with a single GTX 1080 Ti.

Table 4 Character verification without CNN output score refinement

| IC Number | Misjudged Images | Passing Rate (%) | Average Execution Speed (ms) |
|-----------|------------------|------------------|------------------------------|
| IC001 | 63 | 94.81 | 69 |
| IC002 | 35 | 97.52 | 81 |
| IC003 | 28 | 97.77 | 92 |
| IC004 | 4 | 99.76 | 62 |
| IC005 | 953 | 52.28 | 67 |
| IC006 | 905 | 54.50 | 59 |
| IC007 | 807 | 59.45 | 57 |
| IC008 | 1050 | 47.32 | 61 |
| IC009 | 780 | 37.60 | 72 |
| IC010 | 661 | 46.30 | 74 |
| Average | 529 | 69.73 | 69 |

Table 5 Character verification with CNN output score refinement

| IC Number | Misjudged Images | Passing Rate (%) | Average Execution Speed (ms) |
|-----------|------------------|------------------|------------------------------|
| IC001 | 10 | 99.18 | 245 |
| IC002 | 8 | 99.43 | 271 |
| IC003 | 25 | 98.01 | 341 |
| IC004 | 4 | 99.76 | 238 |
| IC005 | 24 | 98.80 | 236 |
| IC006 | 23 | 98.84 | 217 |
| IC007 | 23 | 98.84 | 222 |
| IC008 | 32 | 98.39 | 220 |
| IC009 | 18 | 98.56 | 251 |
| IC010 | 17 | 98.61 | 266 |
| Average | 18 | 98.84 | 250 |

Table 6 Results of different methods

| Method | Passing Rate (%) | Average Execution Speed (ms) | Training Time |
|-----------------|------------------|------------------------------|---------------|
| Shi et al. [8] | 63.41 | 176 | 3~4 h |
| SSD [9] | 71.68 | 110 | 5~6 h |
| YOLO [14] | 86 | 35 | 3 h |
| AOI Machine | 79.3 | 60 | 1~2 h |
| Proposed Method | 98.84 | 250 | 3 mins |

5.1 Character verification on IC components

5.1.1 Training process of character verification

In the training process, each character was constructed in a different image type according to its angles (0, 90, 180, or 270

degrees). Characters that looked the same from different angles (for example, “8” looks the same at 0 degrees and 180 degrees) were removed from the type lists. Therefore, 600 characters and 120 types were used in the training data. Each type used approximately 100 characters for the testing data. Table 3 lists the training parameters. Figure 11 presents the learning process and system converge in 10 epochs; the amount of time required to train the data was only 3 min.

5.1.2 Experimental results of character verification

IC001 to IC010 were used in the character-verification experiments. All types of IC components were approximately 1500 images. The images contained approximately 20 characters and symbols. CNN contour border detection was used in this experiment. Table 4 demonstrates that without adding the refinement mechanism, the average misjudged image was 529 photos and the passing rate was only 69.73%. Compare with Table 5, the refinement mechanism is added, the passing rate increases substantially and reached 98.84%. Even the average execution time was longer than before, and it still fits manufacturers’ standards.

Table 6 presents a passing rate comparison of the proposed method with the other methods, Shi et al. [8], SSD [9], YOLO [14] and conventional AOI machine, under same conditions. The high complexity of Shi et al. [8] and SSD [9] required more time to train their deep learning networks. The conventional AOI machine also needed more training time because the parameters of all IC components were manually adjusted. YOLO [14] shows short average execution speed, however many characters missed detection which causes YOLO [14] to have a lower passing rate than the proposed method. The proposed method required less time for training, and it is generalizable because it does not involve the development of new training data, as is the case in other methods.

Fig. 12 displays examples of success from using the proposed method. The left side presents the results of contour

Fig. 12 Examples of success









| | | | |
|--|---|--|------------------------------------|
|  <p>(a)</p> | <p>0, 2ZE12 0, D9QLJ</p> |  <p>(b)</p> | <p>90, AN5276 90, 645S6E01</p> |
|  <p>(c)</p> | <p>180, TOSHIBA 180, TB9081FG 180, JAPAN1530ESI</p> |  <p>(d)</p> | <p>270, 3IE11 270, D9QNS</p> |

Fig. 13 Examples of misjudgment

| | | | |
|--|---------------------------------|--|-------------------|
|  <p>(a)</p> | <p>0, 59038 0, 4213</p> |  <p>(c)</p> | <p>270, B07C</p> |
|  <p>(b)</p> | <p>0, 77762-1 0, 7924A1</p> |  <p>(d)</p> | <p>180, 9SIIJ</p> |

border detection, and the right side displays the degrees and the strings, which are separated by a comma.

Figure 13 displays some misjudged examples, most of which were caused by blurry characters, noise, or failure of the score to reach the threshold. For example, the dots were not detected in Fig. 13 (a) and (b), the number “4” was verified as “A” in Fig. 13 (b), the number “8” was identified as “B” in Fig. 13 (c), and the score failed to reach the threshold in Fig. 13 (d), indicating that the character fracture was not connected, as was the case with the character “H.”

5.2 Image classification on IC components

5.2.1 Training process of image classification

Six classes of IC images, numbered IC011 to IC016, were used in the experiment. Each class of image was classified in four directions. Each class had approximately 700 images. First, the black pixels were padded to ensure that the images remained the same size. The images were then adjusted to 250×250 for training. In total, 10% of the training data was used as the testing data. Table 7 lists the parameters used in training, and the learning graph is displayed in Fig. 14. The total training time was 45 min.

5.2.2 Experimental results of image classification

Table 8 lists the passing rates of six classes of IC images. The average passing rate was 99.48%, and the average execution time for each IC image was 23.2 ms.

Many deep learning approaches present good performance in image classification such as VGGNet [15], GoogLeNet

Table 7 Classification parameters of IC component images

| Training Epochs | Batch Size | Optimizer Type | Base Learning Rate |
|-----------------|------------|----------------|--------------------|
| 50 | 64 | Adam [13] | 0.0005 |

[16], AlexNet [17], and ResNet [18]. In very recent algorithms, a new pooling technique that combines two consecutive convolutional layers as a pooling operation, was proposed by Liu et al. [19]. It used convolutional layers from a DCNN at first then applied the pretrained CNN on densely sampled image regions and treated the fully-connected activations of each image region as a convolutional layer’s feature activations. Then, another convolutional layer was trained on top of that as the pooling guidance convolutional layer. To improve recognition accuracy and decrease the parameters needed in CNN, Zhang et al. proposed the hybrid model CNN-GRNN [20], which extracted features using CNN then classified images with GRNN which has only one variable and no need to iterate. Another more discriminative feature coding network is designed by Chen et al. called LSO-VLADNet [21]. Expanding the NetVLAD model, an end-to-end feature coding network, LSO-VLADNet, is able to be jointly trained with a deep convolutional neural network for visual recognition. In addition, the feature coding method is the core component of this framework, which links feature extraction and feature pooling, and greatly influences the image classification performance.

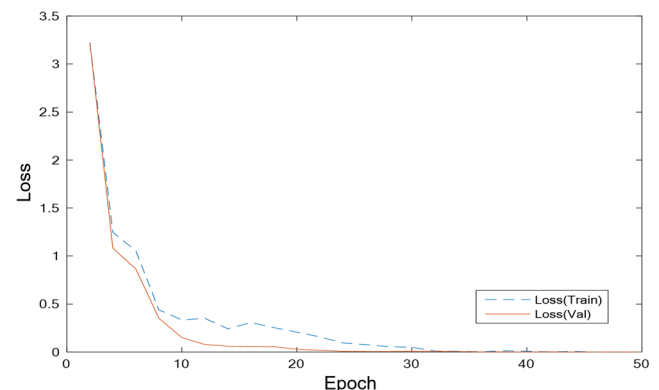
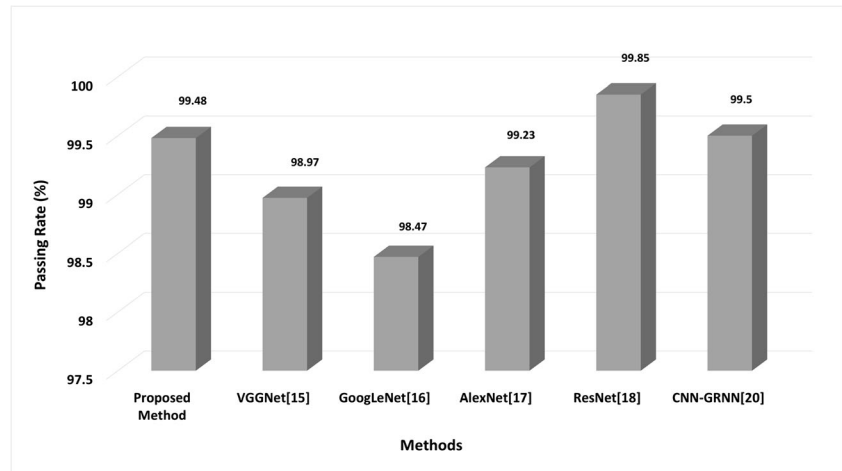


Fig. 14 Learning graph of classification on IC component images

Table 8 Passing rate of IC component image classification

| IC Number | Misjudged Images | Passing Rate (%) | Average Execution Speed (ms) |
|-----------|------------------|------------------|------------------------------|
| IC011 | 9 | 99.46 | 25 |
| IC012 | 5 | 99.67 | 21 |
| IC013 | 15 | 99.14 | 25 |
| IC014 | 7 | 99.63 | 23 |
| IC015 | 6 | 99.56 | 26 |
| IC016 | 11 | 99.41 | 19 |
| Average | 8.83 | 99.48 | 23.2 |

Fig. 15 Passing rate comparison of various methods**Table 9** Training time comparison of various methods

| Methods | Proposed Method | VGGNet [15] | GoogLeNet [16] | Alexnet [17] | ResNet [18] | CNN-GRNN [20] |
|---------------|-----------------|-------------|----------------|--------------|-------------|---------------|
| Training Time | 45 min | 2hr10min | 1h3min | 53 min | 4hr30min | 1 hr |

The evaluation indexes consist of the passing rate and the training time. Figure 15 presents the passing rates of various methods using the same training data and testing data for comparison. The epoch was set to 50, and the optimizer type was set to Adam [13].

According to the experimental results in Fig. 15, the passing rate of the proposed method is better than those of other methods, except the ResNet [18] and CNN-GRNN [20]. Even if ResNet [18] shows the highest passing rate in all the methods, its training time reveals obvious differences as shown in Table 9. Also, in Table 10, the execution speed and loading time per image show efficient advantages of the proposed method. CNN-GRNN also reveals better passing rate than the proposed method, however, CNN and GRNN networks both need to be trained. This causes the training procedure to

be more complex and have a longer training time. CNN-GRNN might cause tedious work if employed by a manufacturer. Therefore, the proposed method not only reduces the amount of required time but also is accessible and flexible for manufacturers.

Table 10 Execution speed and loading time comparison of ResNet and proposed method

| | Execution Speed (Frames/s) | Loading Time (ms/image) |
|-----------------|----------------------------|-------------------------|
| ResNet [18] | 7.19 | 139 |
| Proposed Method | 35.71 | 28 |

6 Conclusions

The proposed deep-learning methods were used in PCB testing, which involved character verification on IC components and classification on IC component images. IC component character verification employed contour border detection with CNN and then used the refining output score from CNN to increase accuracy. IC component image classification employed a different CNN structure and same refinement mechanism to increase accuracy.

According to the experimental results, the passing rates of both methods reached 98.84% and 99.48%, and the times required for training were less than those of other methods. Both methods met manufacturer requirements and have been implemented on the production lines. In future works, the program will automatically test an image again after it has been misjudged. The program will be embedded in machines to shorten processing time after fetching massive images from cameras.

References

1. Cho HJ, Park TH (2008) Template matching method for SMD inspection using discrete wavelet transform. Proc SICE Annual Conference: 3198–3201
2. Crispin AJ, Rankov V (2007) Automated inspection of PCB components using a genetic algorithm template-matching approach. Int J Adv Manuf Technol 35(3–4):293–300
3. Lee JJ, Lee PH, Lee SW, Yuille A, Koch C (2011) AdaBoost for text detection in natural scene. International Conference on Document Analysis and Recognition: 429–434
4. Epshtein B, Ofek E, Wexler Y (2010) Detecting text in natural scenes with stroke width transform. IEEE Computer Society Conference on Computer Vision and Pattern Recognition: 2963–2970
5. Nava CF, Gonzalez FF (2015) OCR for unreadable damaged characters on PCBs using principal component analysis and bayesian discriminant functions. Proceedings international conference on computational science and computational intelligence (CSCI 2015): 535–538
6. Li W, Neullens S, Breier M, Bosling M, Pretz T, Merhof D (2014) Text recognition for information retrieval in images of printed circuit boards. Annual conference of the IEEE industrial electronics society (IECON 2014): 3487–3493
7. Zhang S, Liu Y, Jin L, Luo C (2017) Feature enhancement network: a refined scene text detector. Association for the Advancement of artificial intelligence (AAAI 2018): 2612–2619
8. Shi B, Bai X, Yao C (2017) An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. IEEE Trans Pattern Anal Mach Intell 39(11): 2298–2304
9. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, Berg AC (2016) SSD: single shot multibox detector. European conference on computer vision (ECCV2016): 21–37
10. Liao M, Shi B, Bai X, Wang X, Liu W (2017) Textboxes: a fast text detector with a single deep neural network. Association for the Advancement of artificial intelligence (AAAI 2017): 4161–4167
11. Otsu N (1979) A threshold selection method from gray-level histograms. IEEE Trans Syst Man Cybern 9(1):62–66
12. Suzuki S, Be K (1985) Topological structural analysis of digitized binary images by border following. Comput Vision, Graph Image Process 30(1):32–46
13. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980
14. Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: unified, real-time object detection. Proc IEEE Conf Comput Vision and Pattern Recogn, Las Vegas, Nevada, USA: 779–788
15. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv: 1409.1556
16. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition: 7–12
17. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. Proc 25th Int Conf Neural Inform Process Syst 1:1097–1105
18. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition, Las Vegas, Nevada, USA: 770–778
19. Liu L, Shen C, Heng AVD (2017) Cross-convolutional-layer pooling for image recognition. IEEE Trans Pattern Anal Mach Intell 39(11):2305–2313
20. Zhang J, Shao K, Luo X (2018) Small sample image recognition using improved convolutional neural network. J Vis Commun Image Represent 55:640–647
21. Chen B, Li J, Wei G, Ma B (2018) A novel localized and second order feature coding network for image recognition. Pattern Recogn 76:339–348
22. Chen PH, Si S, Kumar S, Li Y, Hsieh C (2018) Learning to screen for fast Softmax inference on large vocabulary neural networks. arXiv preprint arXiv: 1810.12406

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Chun-Hui Lin received the M.S. degree in Computer Science from the University of Texas at Dallas, Texas USA, in 2017. Currently, she is a Ph.D. student in Computer Science and Information Engineering at National Cheng Kung University, Tainan, Taiwan. Her research interests are image processing, intelligent control and machine/deep learning.



Shyh-Hau Wang received B.S. degree in biomedical engineering from Chung Yuan Christian University (CYCU), Chung Li, Taiwan, in 1986; M.S. degrees in biomedical engineering and electrical engineering from Drexel University, Philadelphia, PA, in 1992; and Ph.D. degree in bioengineering from The Pennsylvania State University (PSU), University Park, PA, in 1997. He worked as research associate at the Department of Biomedical Engineering, University of

Virginia, in 1997 and then as postdoctoral fellow at the NIH Ultrasound Transducer Resource Center, PSU, in 1998. Subsequently, Dr. Wang served for the Department of Biomedical Engineering, CYCU, as assistant professor (1998), associate professor (2001), and professor (2006). Since 2009, he joined the Department of Computer Science and Information Engineering & Institute of Medical Informatics at National Cheng Kung University, Tainan, Taiwan. His research interests are in the areas of biomedical ultrasound imaging, ultrasound tissue/material characterization, signal/image processing, medical instrumentation and informatics, and machine/deep learning.



Cheng-Jian Lin received the Ph.D. degree in electrical and control engineering from the National Chiao-Tung University, Taiwan, R.O.C., in 1996. Currently, he is a Lifetime Distinguished Professor of Computer Science and Information Engineering Department, National Chin-Yi University of Technology, Taichung County, Taiwan, R.O.C. His current research interests are computational intelligence, intelligent transportation

system, intelligent control, image processing, and mobile robotic control.