Machine Vision and Applications (2021) 32:66
https://doi.org/10.1007/s00138-021-01192-8

ORIGINAL PAPER

TsanKit: artificial intelligence for solder ball head-in-pillow defect inspection

Ting-Chen Tsan[1] · **Teng-Fu Shih**[1] · **Chiou-Shann Fuh**[1]

Received: 21 August 2019 / Revised: 6 August 2020 / Accepted: 5 March 2021

Abstract
In this paper, we propose an AI (Artificial Intelligence) solution for solder ball HIP (Head-In-Pillow) defect inspection. The HIP defect will affect the conductivity of the solder balls leading to intermittent failures. Due to the variable location and shape of the HIP defect, traditional machine vision algorithms cannot solve the problem completely. In recent years, Convolutional Neural Network (CNN) has an outstanding performance in image recognition and classification, but it is easy to cause overfitting problems due to insufficient data. Therefore, we combine CNN and the machine learning algorithm Support Vector Machine (SVM) to design our inspection process. Referring to the advantages of several state-of-the-art models, we propose our 3D CNN model and adopt focal loss as well as triplet loss to solve the data imbalance problem caused by rare defective data. Our inspection method has the best performance and fast testing speed compared with several classic CNN models and the deep learning inspection software SuaKIT.

Keywords Solder ball · Head-In-Pillow · Defect inspection · Artificial intelligence · Convolutional neural networks

1 Introduction

1.1 Overview

In manufacturing, it is important to maintain product quality. Manufacturing quality control can be defined as the oversight of aspects of manufacturing production. The goal of implementing manufacturing quality control is to produce products that conform to industry, company, and consumer expectations every time. On the production line, inspection [1] is essential to controlling the quality of products by helping to fix the sources of detected defects immediately, and it is useful to improve productivity, reduce defect rates, rework, and waste.

With the development of technology, the demand for automated and intelligent inspection is gradually growing to reduce labor costs, especially for semiconductor processes and Integrated Circuit (IC) packaging. The precise architecture of electronic components makes it difficult to observe most defects through human eyes, so some technical methods are used to inspect the products, such as Automated X-ray Inspection (AXI), Automated Optical Inspection (AOI), and Solder Paste Inspection (SPI).

In IC packaging, the Printed Circuit Board (PCB) plays an important role in many electronic products. A Ball Grid Array (BGA) [2] is a surface mount chip package that uses a grid of solder balls (also known as solder bumps) to connect the PCB. PCB can be single-sided, double-sided, or multi-layer; for multi-layer PCB, many overlapped electronic components make the defect inspection more difficult and challenging.

Due to the complexity of most images of electronic components, traditional machine vision methods cannot solve the problem completely. In recent years, deep learning has been widely used in many computer vision tasks. In particular, Convolutional Neural Network (CNN) has an outstanding performance in image object recognition and classification. Different levels of features can be integrated by the deep network structure. Finally, the complicated high-level features can be combined with an end-to-end network to predict the result.

Although the performance of deep learning is remarkable, the trained models will easily over-fit due to insufficient data. Therefore, we try to combine machine learning algorithms to further compute the uncertainty level and increase accuracy.

✉ Teng-Fu Shih
g104018004@gmail.com

[1] National Taiwan University, Taipei 106, Taiwan

Published online: 24 March 2021

Another difficulty is that it is hard to obtain defective data because defective products would need to take corrective action on the production line immediately. This will lead to the problem of data imbalance, and it can make the model blindly learn the characteristics of normal data while ignoring defective data.

In this paper, our goal is to develop a solder ball Head-In-Pillow (HIP) defect inspection algorithm by combining deep learning with machine learning. We aim to solve the data imbalance problem through Artificial Intelligence (AI) techniques. We will also compare the performance and execution time of our method with several classic CNN models and the deep learning inspection software SuaKIT.

1.2 Head-in-Pillow defect

Head-In-Pillow (HIP) [3] is a latent solder ball defect occurring in the soldering process. Once formed in the solder joints, HIP defects often escape inspection and tests on the factory floor as there may still be mechanical and electrical contact. However, the defect will cause the unstable conductivity of the particular BGA balls leading to intermittent failures. It is difficult to achieve zero miss detection rate of the HIP defect without experts or field application engineers' inspection. So far, HIP defect inspection is still a challenging issue.

It is hard to find the location of HIP defects from 2D X-Rays images due to the variable shape of the defect. We try to reconstruct the 3D solder ball model from different angles of 2D X-Rays images. The 3D solder ball model can not only provide more information but also represent the location of HIP defects more clearly.

1.3 Acquisition of PCB images

Before we perform defect inspection, we have to acquire the images of PCB. X-rays are used due to the characteristic of penetration and the absorption ability of different materials. Computed Laminography (CL) [4] is an image forming method of X-ray testing that yields slice images of an object by linear translation or rotation of the object relative to the tube-detector system. Laminography is based on the relative motion of the X-ray source, the detector, and the object. Points from a particular plane called "focal plane" will be projected at the same location on the detector, and all other structures of the object above and below the focal plane will be projected during the movement at different locations on the detector.

The X-ray projected images are obtained by taking several photographs from different angles. With these X-ray images, we can reconstruct and observe the cross section of different levels noninvasively without breaking the PCB.

Many algorithms are proposed for tomographic reconstruction which is mainly based on the mathematics of the Radon Transform, for example, Algebraic Reconstruction Technique (ART), Simultaneous Algebraic Reconstruction Technique (SART), and Filtered Back Projection (FBP).

Under the influence of X-rays, there is a large amount of white noise in the projected images. Therefore, we need to optimize the images first before reconstruction; otherwise, we will get inferior quality images. White noise is removed by averaging multiple images over time.

2 Related works

2.1 Cause of Head-in-Pillow defect

Head-In-Pillow (HIP) is one of the most common failure modes in BGA board assembly including System-In-Package (SIP) and Package on Package (PoP) [3]. The HIP is formed because the solder ball could not be melted uniformly with the solder on the PCB pad during reflow. The reasons for HIP defects are as follows:

- an air gap between the two solder balls;
- surface oxidation or contamination of solder balls;
- poor wetting of the solder (solder temperature too low, insufficient melting time, poor flux, …);
- distortion caused by the heat of the tin stove.

For either case, the root cause is that the solder ball is not in contact all the time with the soldering pads throughout the reflow processes.

The PoP has two parts, including top and bottom BGAs. During the reflow, the gap between the solder ball and the soldering pad on PCB is created mainly due to the BGA warpage caused by the high temperature. There are typically four cases for the warpage shown in Fig. 1. In Cases 1 and 2, the warpage of the top and bottom BGAs is compensated by each other. The warpage in Case 3 is the worst situation. It could create a wider gap and has a greater probability to cause the HIP defect on the periphery of the BGA. The magnitude of the warpage in Case 4 is constrained, and it creates fewer HIP defects for the outer row.

The temperature will gradually decrease after reflow. At the same time, the BGAs gradually restore to the state before the deformation. However, the temperature at this time is lower than the melting point of the solder balls and the solder paste. That is, the states of the solder balls and the solder paste have already changed from the molten state to the solid-state and cannot be blended perfectly. Finally, the solder balls and solder paste are in contact with each other again and form a shape similar to a head resting on a pillow,

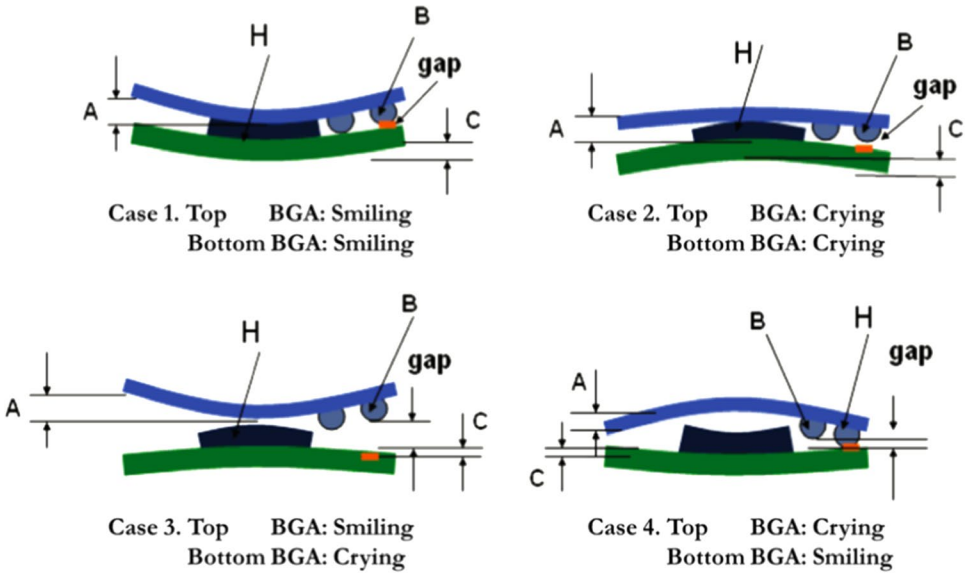

Fig. 1 The schematic chart of typical warpage patterns of the top and bottom BGAs [25]

which is the HIP defect. If the gap is still present and large enough, an open solder defect may be formed.

The warpage of package and PCB, solder paste printing, flux type, and solder alloy all play leading roles in a HIP formation. There are some measures recommended by [25] to reduce HIP defects:

- minimize BGA warpage during reflow;
- control the warpage profile of PCB and bottom BGA;
- use robust flux;
- increase solder paste volume for SIP;
- increase BGA ball size and coplanarity.

2.2 Inspection for Head-in-Pillow defect

Most of the current methods for HIP defect inspection are based on the 2D X-ray images. According to the geometric characteristics of the normal and defective solder balls, several inspection methods are proposed by electronics manufacturers and inspection system providers in the world, such as Test Research Inc., Agilent Technologies, Flextronics, and ViTrox [5–7].

For BGA joints, we can obtain three slices: ball slice, at the center of the BGA; chip slice, near the package level pad; and pad slice, near the board level pad shown in Fig. 2 [7]. HIP usually occurs at the pad slice. We can distinguish normal and defective solder balls according to the area or the circularity of the slices and check the difference between the area of an individual ball and its corresponding neighbor.

- **Area**: Compared with a normal solder ball, the area of the chip slice of a HIP solder ball is larger, and the area of the pad slice is smaller.
- **Circularity**: Compared with a normal solder ball, the shape of the pad slice of a HIP solder ball is non-circular.

3 Background

Artificial Neural Networks (ANNs) and Support Vector Machine (SVM) [8] are important models and algorithms of machine learning.

Deep learning [9] is a class of machine learning techniques that exploit many layers of nonlinear information processing for supervised or unsupervised feature extraction and transformation. Among different deep learning structures, Convolutional Neural Network (CNN) plays an important role and is successfully applied to the field of computer vision, such as image classification, image segmentation, and object detection. The following sections will briefly introduce the principle of CNN and some of the most representative models.

Convolutional Neural Network (CNN) [10] is inspired by the organization of the visual cortex. The convolution emulates the response of the cortical neuron to visual stimuli. Therefore, CNN is designed as a shared-weight neural network combined with a fully connected neural network.

LeNet-5 [11] is an early classical CNN model used for hand-written digit recognition. The architecture of LeNet-5 is simple but contains all the important components of CNN.

Since the breakthrough of AlexNet in 2012, most popular CNN models just stack convolution layers deeper and wider, which means that the numbers of layers and neurons keep increasing. Visual Geometry Group (VGG) [12] is a classic model that proves that increasing the depth of the network can improve the accuracy of image recognition.

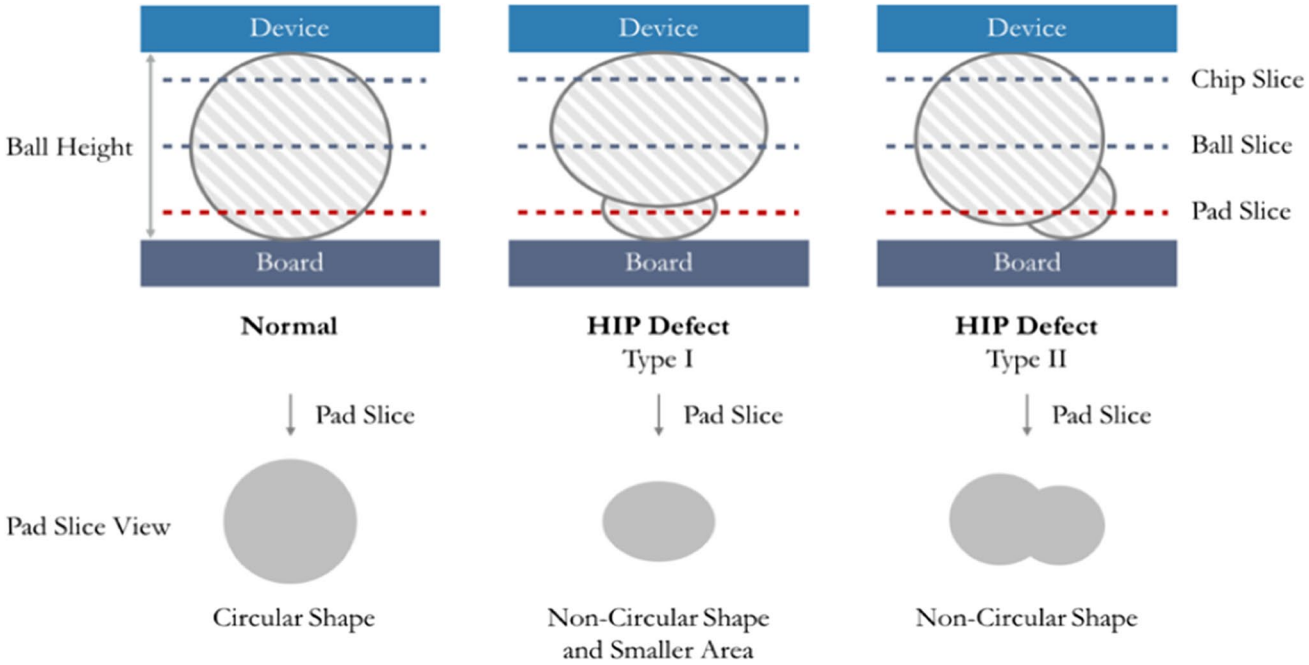

Fig. 2 Comparison of three slices for BGA joints between the normal and HIP solder balls

However, due to very deep networks and large convolution operations, there are some problems with overfitting and expensive computations. This situation continued until GoogLeNet [13, 14] was proposed in 2014. To optimize the feature extraction module and increase network depth and width while reducing parameters, Inception module was born.

Although increasing the depth of the network can improve the accuracy of the recognition, it may cause the vanishing gradient problem when the network is too deep. Residual Neural Network (ResNet) [15] is a solution to the vanishing gradient problem. ResNet proposed a residual block with shortcut connections. The input not only performs the convolution processes but also adds to the output in each residual block. This way carries important information in the preceding layer to the following layers.

Dense Convolutional Network (DenseNet) was proposed in 2017 [16]. DenseNet consists of dense blocks and transition layers. In the dense block, each layer receives additional inputs from all preceding layers and passes its own feature maps to all subsequent layers. The feature maps in a dense block have the same size, so they can be connected through channel-wise concatenation. The transition layer connects two adjacent dense blocks and reduces the size of the feature maps by pooling.

Due to the dense connection, DenseNet improves the back-propagation of the gradient, making the network easier to train. Same as ResNet, DenseNet can avoid some important features lost in the deep network and alleviate the vanishing gradient problem. Additionally, DenseNet achieves feature reuse to strengthen feature propagation and improve efficiency. The parameters of DenseNet are less than ResNet and get better performance on CIFAR-100 and ImageNet.

4 Methodology

4.1 Overview

Our detailed methods for solder ball HIP defect inspection are described in here. The whole workflow is shown in Fig. 3.

4.2 3D Solder balls extraction

There are many electronic components on the PCB, and solder balls are typically present in layers near the center of the PCB.

After obtaining the 3D PCB images [17], we have to extract the solder balls on the PCB for further inspection. The flowchart of 3D solder ball extraction is shown in Fig. 4.

To avoid the influence of other electronic components, first of all, we need to automatically crop the Region Of Interest (ROI) according to Computer-Aided Design (CAD) solder ball coordinates to only include solder balls and ignore the blurred and incomplete solder balls. The result is shown in Fig. 5.

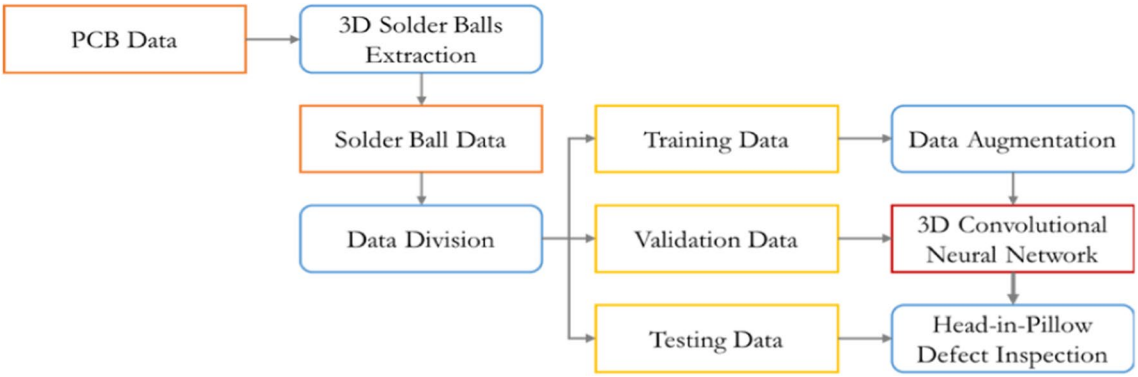

Fig. 3 The workflow for solder ball HIP defect inspection

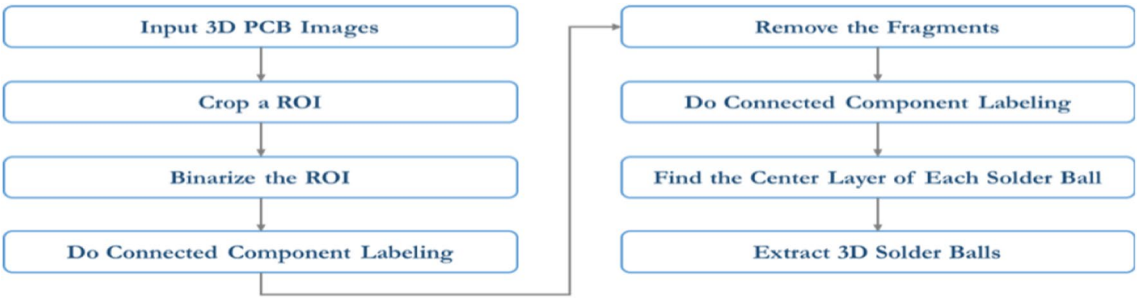

Fig. 4 The flowchart of 3D solder ball extraction

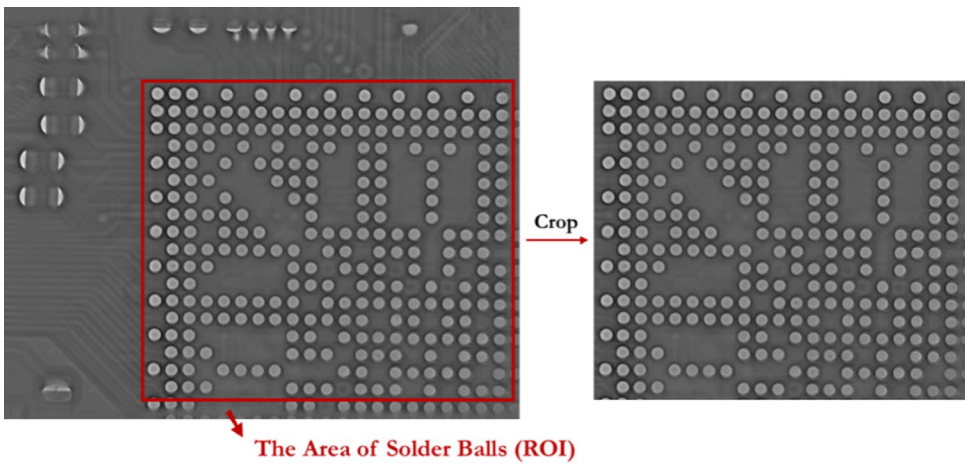

Fig. 5 The illustration of cropping the area of solder balls. The red box area represents the ROI including solder balls

Next, we binarize the area of solder balls by Otsu thresholding [18] and use Connected-Component Labeling (CCL) [19] to detect the solder balls. Due to some background noise and incomplete solder balls on the image boundary, the fragments are removed according to the number of each label (region ID or ball ID). After automatically cropping the Region of Interest (ROI) according to Computer-Aided Design (CAD) solder ball coordinates, CCL is used again to get the labels and area of real solder balls. After Connected-Component Labeling, we can find the largest cross section of each ball.

Since the PCB is slightly tilted (about 8 pixels of height difference), the center of each solder ball is in different layers. If we use the center layer of the entire PCB to extract the solder balls, we may get top–bottom unbalanced and thus incomplete balls shown in Fig. 6a. Therefore, according to the number of labels in different layers, we need to automatically compute the center layer corresponding to the largest

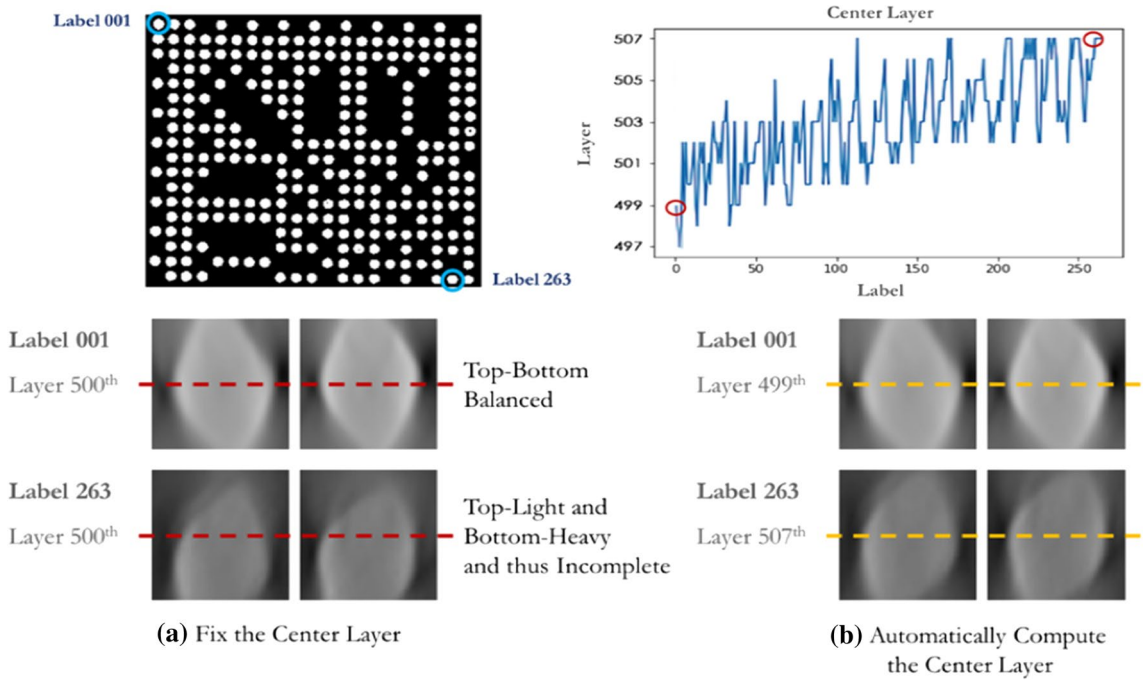

Fig. 6 **a** The slice images of 3D solder balls with fixed center layer. **b** The slice images of 3D solder balls with automatically computed center layers for different solder balls

cross section of each ball, and then, we can extract the 3D solder balls. The extraction results are shown in Fig. 6b.

4.3 Data augmentation

After obtaining the 3D solder ball data, we divide the dataset into training data, validation data, and testing data. Since the amount of training data (345 solder balls: 315 normal solder balls, 30 defective solder balls) is not enough, we create more samples with data augmentation and deal with normal data and defective data separately.

There are three sets of PCB data in Fig. 7. The first set is reconstructed with 128 optimized projected images and thus has the best image quality. The second set is reconstructed with 16 optimized projected images and has medium image quality. The third set reconstructed with 16 images that are not optimized to remove white noise for fast processing, so the data in this set have the worst image quality. Through observing the slice images, we can find that the data in the

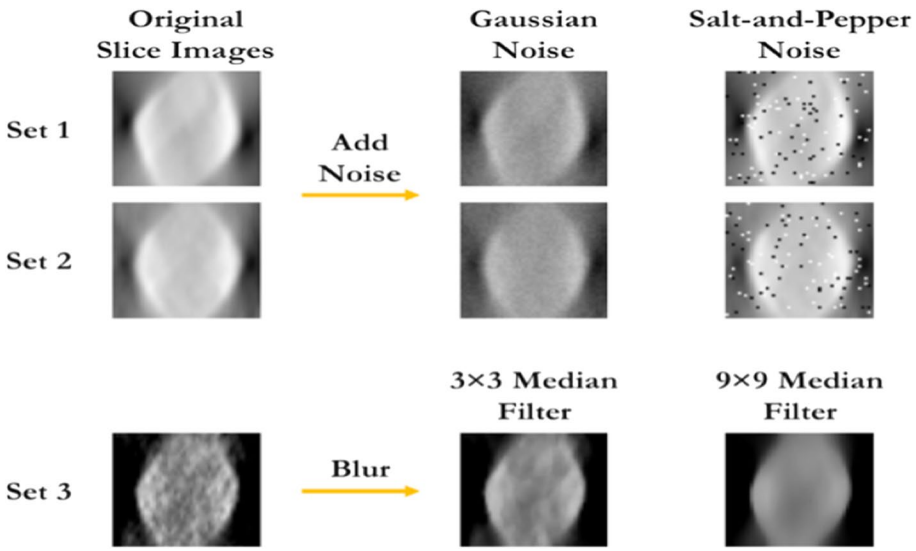

Fig. 7 The slice images of the 3D normal data in three sets of PCB data and the results after data augmentation (adding noise and blurring to 3D data)

first and second sets are smoother than the data in the third set. Due to the different characteristics of the PCB, we will take different augmentation methods.

For the normal data, to improve noise robustness of the model, we add Gaussian noise and salt-and-pepper noise to the data in the first and second sets and blur the data in the third set to remove noise by median filters with 3×3 and 9×9 kernels. We doubled the amount of normal data.

Compared with the number of normal solder balls (315), the number of defective solder balls (30) is relatively insufficient. Therefore, for defective data, in addition to adding noise or blurring, we perform several geometric transformations including rotation and reflection. We increase the amount of defective data by seventeen times.

4.4 3D Convolutional neural network

The state-of-the-art CNN models (VGG-16, GoogLeNet Inception v1, ResNet-50, and DenseNet-121) have a significant impact on the domain of deep learning, especially in image classification [13–16]. Referring to the advantages of these models, we propose our 3D CNN model shown in Fig. 8.

In our model, there are four dense connection blocks and each dense connection block includes two convolutional blocks shown in Fig. 9. It can strengthen feature propagation and avoid losing some information in the deep convolutional path. Since the feature maps of a dense connection block will have multiple growths, we use $1\times1\times1$ convolution kernels with stride 2 to reduce the dimension of the feature maps. The $1\times1\times1$ convolution kernels can compress the features and keep the useful information well. Additionally, the global average pooling is used not only to preserve the spatial information extracted from previous layers but also to normalize the input tensor to the same size. Therefore, our model can adapt to the different sizes of input.

In the convolutional block, there are three convolutional paths shown in Fig. 10. The first path includes a single 3D convolutional layer with $3\times3\times3$ kernel; the second path includes a 3D average pooling layer with $3\times3\times3$ kernel and a 3D convolutional layer with $1\times1\times1$ kernel; the third path includes a 3D max pooling layer with $3\times3\times3$ kernel and a 3D convolutional layer with $1\times1\times1$ kernel. Finally, the results of each convolutional path are concatenated. Various features can be learned through these different paths, so

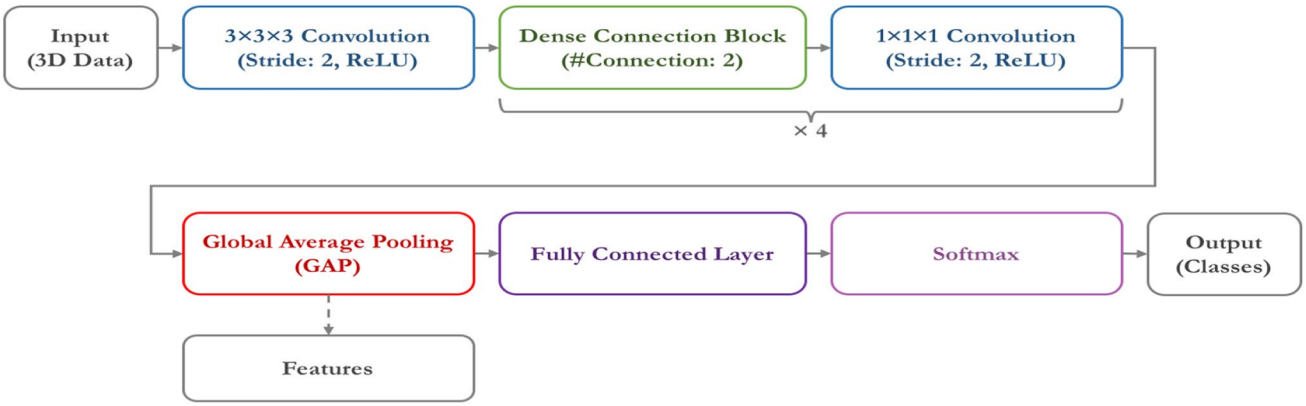

Fig. 8 The architecture of our 3D CNN model. The depth of the model is 14

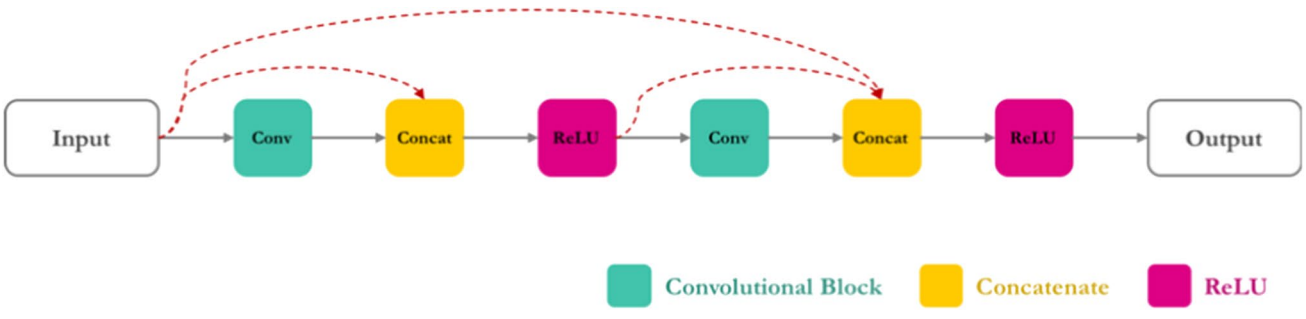

Fig. 9 The dense connection block. The result of the current convolutional block will concatenate with the result of the previous one, and the output will do the Rectified Linear Unit (ReLU) activation function before putting it to the next convolutional block

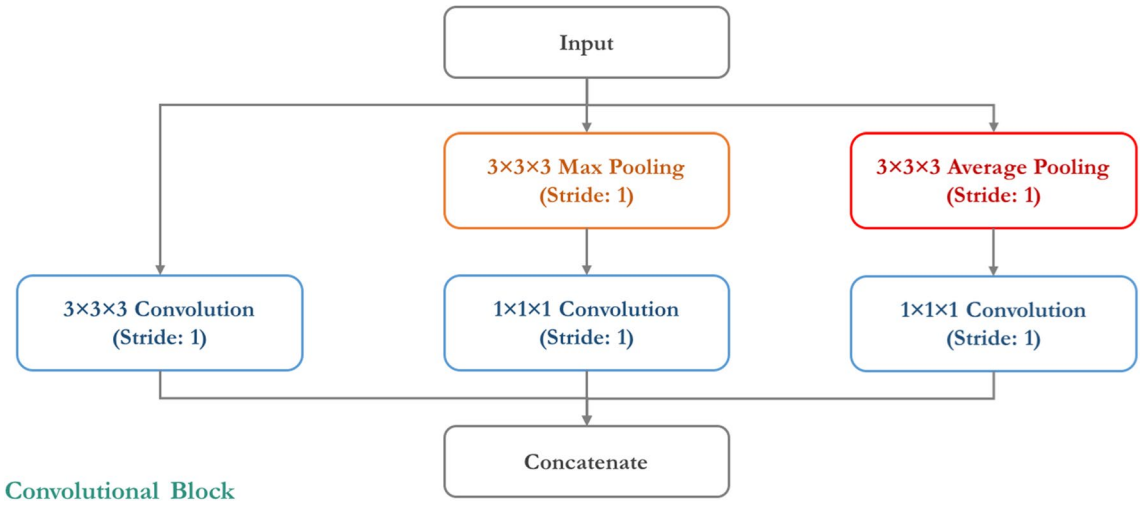

Fig. 10 The convolutional block including three paths

the model can choose better features and be more robust in testing.

4.5 Loss function

4.5.1 Cross-entropy loss

The general loss function for classification problems is Cross-Entropy (CE) loss. Cross-entropy describes the distance between two probability distributions defined as:

$$\mathrm{CE} = -\sum_{i}^{C} \hat{y}_i \log\left(p_i\right) \tag{1}$$

where $p = (p_1,\ldots,p_C)$ is the predicted results; and $\hat{y} = (y_1,\ldots,y_C)$ is the ground truth labels.

The numbers between positive and negative samples vary greatly. It causes a data imbalance problem when training the model and may make the model incapable of identifying positive samples. Therefore, we use a novel loss named "Focal Loss" proposed by [20] to solve the problem. On the other hand, to avoid these few defective data being too sparse in the spatial distribution and separate the distance between the positive and negative samples, we add the triplet loss in cosine similarity to aid in model learning.

4.5.2 Focal loss

A general method for addressing class imbalance is to introduce a weighting factor $\alpha \in [0, 1]$ for Class 1 (Defect) and $1 - \alpha$ for Class 0 (Normal). The α-balanced CE loss is a simple extension to CE loss defined as:

$$\mathrm{CE}\left(\alpha_t,\, p_t\right) = -\alpha_t \log\left(p_t\right) \quad \text{where} \quad \alpha_t = \begin{cases} \alpha, & \text{if } \hat{y} = 1 \\ 1-\alpha & \text{if } \hat{y} = 0 \end{cases} \tag{2}$$

$\alpha = 0.5$ when the number of positive and negative samples is equal; $\alpha < 0.5$ when the number of positive samples is more than negative samples; otherwise, $\alpha > 0.5$.

Based on α-balanced CE loss, Focal Loss (FL) adds a modulating factor $(1 - p_t)^{\gamma}$ to the standard CE loss defined as:

$$\mathrm{FL}\left(p_t\right) = -\left(1 - p_t\right)^{\gamma} \log\left(p_t\right) \tag{3}$$

where $\gamma \geq 0$. When $\gamma = 0$, FL is equivalent to CE; as γ is increased, the effect of the modulating factor is likewise increased. The modulating factor reduces the loss contribution from easy examples, and it can increase the importance of correcting misclassified examples.

In practice, an α-balanced variant of the focal loss is used:

$$\mathrm{FL} = -\hat{y}\alpha(1-p)^{\gamma}\log(p) - (1-\hat{y})(1-\alpha)p^{\gamma}\log(1-p) \tag{4}$$

where $\alpha = 0.25$ and $\gamma = 2$ work best in [21]. Since the positive samples (Defects) are less than the negative samples (Normal), we set $\alpha = 0.75$ and $\gamma = 2$ in our experiments.

4.5.3 Triplet loss in Cosine similarity

Referring to the concept of triplet loss for few-shot learning and face recognition [22], we hope to reduce the distance between data in the same class and increase the distance between the different classes of data. The triple includes an anchor sample A, a positive sample P, and a negative sample N. The anchor sample is selected randomly from the training samples; the positive and negative samples belonging to

the same class and different class as the anchor are selected from the training samples at random. The loss maximizes the cosine similarity between the anchor and the positive and minimizes the cosine similarity between the anchor and the negative.

4.5.4 Our loss

The features extracted from multi-layer convolution are used to calculate the cosine similarity between different data. The cosine similarity between the anchor features and the positive/negative features is described as:

$$\text{similar}(x, \hat{x}) = \frac{\sum_i x(i) \cdot \hat{x}(i)}{x\hat{x}} \tag{5}$$

where x is the anchor features; $\hat{x}$ is the positive features x_p or the negative features x_n; $x(i)$ is the i-th dimensional feature. The triplet loss in cosine similarity is defined as:

$$\text{TL}_{\cos} = \max\left(m - \text{similar}\left(x, x_p\right) + \text{similar}\left(x, x_n\right), 0\right)$$

where m is the margin between different classes.

Finally, we combine the focal loss and the triplet loss in cosine similarity as the loss function of our model:

$$\text{Loss} = \text{FL} + \lambda \text{TL}_{\cos} \tag{6}$$

where λ is the scale factor.

4.6 Inspection process

There are only hundreds of data (345) to train our CNN model, and it is far fewer than the amount of data for general deep learning tasks. To avoid the overfitting problem, especially at fully connected layers, we add SVM to aid inspection in the process. The flowchart of the inspection process is shown in Fig. 11.

In the training stage, we input the training data to the CNN model and finally export the model with the best parameters evaluated by the validation data. There are two parts to the testing stage. At first, the testing data are input to the CNN model to obtain the probability of normal class P_0 and the probability of defect class P_1, where $P_0 + P_1 = 1$. Additionally, the features of the training data extracted by the CNN model are used to train the first SVM model. Then,

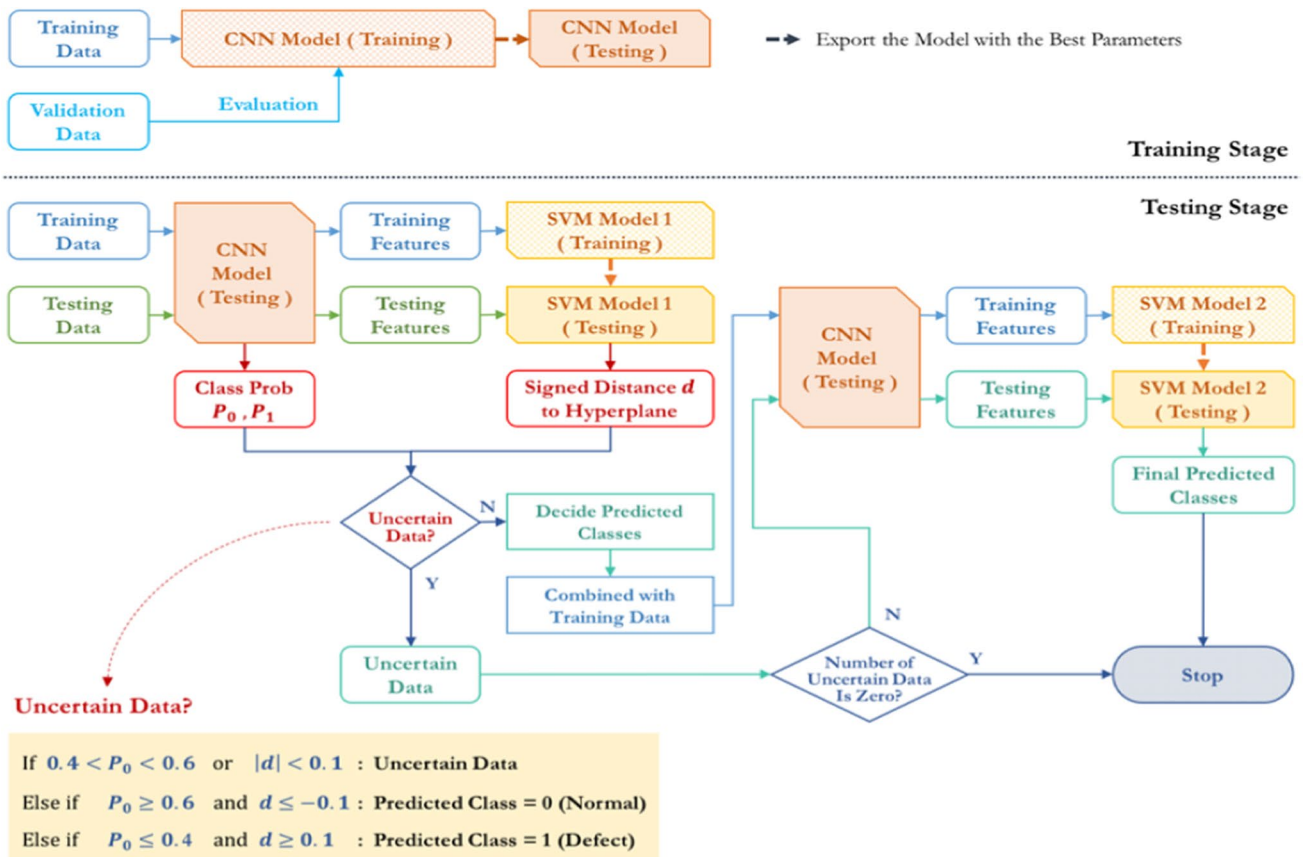

Fig. 11 The flowchart of our inspection process

Table 1 System environment for our experiment

System environment	
Central processing unit	Intel® Core™ i7-8700
Random access memory	32.0 GB
Graphics processing unit	NVIDIA GeForce GTX 1080 Ti

the features of the testing data will be input to the SVM model to get the signed distance d from each data to the hyperplane.

Generally, for CNN, the predicted class will be Class 0 (Normal) if $P_0 > 0.5$; otherwise, the predicted class will be Class 1 (Defect). For SVM, the predicted class will be Class 0 if $d < 0$; otherwise, the predicted class will be Class 1. Since we consider the results of CNN and SVM at the same time, and the results may be inconsistent, we cannot determine the predicted results. Therefore, we pick out some data as the uncertain data from the testing data to execute the next inspection. The uncertain data are defined as:

$$\begin{aligned} \boldsymbol{U} &= \left\{x | x \in \boldsymbol{S} \text{ and } \left(0.4 < P_0(x) < 0.6 \text{ or } |\mathrm{d}(x)| < 0.1\right)\right\} \\ \boldsymbol{R} &= \boldsymbol{S} - \boldsymbol{U} \end{aligned} \tag{7}$$

where $\boldsymbol{S}$ is the testing dataset, $\boldsymbol{U}$ is the uncertain dataset, $\boldsymbol{R}$ is the remaining dataset, and $d(x)$ is the margin distance of the SVM.

For the remaining data, we have to determine the predicted results defined as:

$$y_{\text{pred}}(x) = \begin{cases} 0, & x \in R \text{ and } \left(P_0(x) \geq 0.6 \text{ and } \mathrm{d}(x) \leq -0.1\right) \\ 1, & x \in R \text{ and } \left(P_0(x) \leq 0.4 \text{ and } \mathrm{d}(x) \geq 0.1\right) \end{cases} \tag{8}$$

where x is the remaining data and y_{pred} is the predicted result. If the number of uncertain data is zero, we finish the inspection; otherwise, we go to the next part. In the second part, the remaining data are combined with the training data to train the second SVM model. Then, we inspect the uncertain data and obtain the final predicted results.

5 Experimental results

5.1 The system environment and datasets

The system environment for our experiment is in Table 1.

There are three sets of 3D PCB data numbered as PCB1, PCB2, and PCB3 in our experiments. PCB1 only contains the top left of a BGA board, and there are 264 solder balls including 247 normal solder balls and 17 defective solder balls in PCB1. PCB2 is composed of four sub-images. There are 1199 solder balls including 1176 normal solder balls and 23 defective solder balls in the sub-images. PCB1 and PCB2 represent the same PCB, and they are reconstructed with 128 projected images and 16 projected images, respectively. The area marked with a yellow box in PCB2 corresponds to PCB1.

PCB3 is reconstructed with 16 projected images and composed of four sub-images. There are 754 solder balls including 732 normal solder balls and 22 defective solder balls. The characteristic of PCB3 is completely different from PCB1 and PCB2 because the projected images are not optimized with temporal image averaging. Additionally, the size of the solder balls in PCB3 ($96 \times 96 \times 96$ pixels) is twice that of PCB1 and PCB2 ($48 \times 48 \times 48$ pixels). The detailed information of each PCB is in Table 2.

There are a total of 2217 solder ball data (2155 normal, 62 defects) in our dataset. We divide the dataset into training data, validation data, and testing data in Table 3.

- Training data: 105 normal data and 10 defective data of each PCB;

Table 2 The information of each PCB

	Resolution (Pixels)	#Projected images	#Sub-image	#Solder balls	Ball size (Pixels)
PCB 1	1472×1176×1000	128	1	264	48×48×48
PCB 2	2277×2840×200	16	4	1199	48×48×48
PCB 3	3426×3428×209	16	4	754	96×96×96

Table 3 The number of normal solder balls and defective solder balls of each PCB

	Normal				Defect				Total
	Train	Valid	Test	Sum	Train	Valid	Test	Sum	
PCB 1	105	44	98	247	10	3	4	17	264
PCB 2	105	44	1027	1176	10	3	10	23	1199
PCB 3	105	44	583	732	10	3	9	22	754
Total	315	132	1708	2155	30	9	23	62	2217

- Validation data: 44 normal data and 3 defective data of each PCB;
- Testing data: the remaining 1708 normal data and 23 defective data.

5.2 Evaluation

The ratio of the defective data to the whole dataset is only 2.8%. It is not comprehensive to evaluate a model only by accuracy in the case of data imbalance. Precision and recall can better represent the performance of the model when the numbers of positive and negative samples vary greatly.

Given a classifier and a set of instances, a Confusion Matrix (CM) [23] can be formed to calculate many common metrics and visualize the performance of the classifier. We compare the performance of different models by observing confusion matrix, precision, and recall. It is quite important to inspect all the defective data, so we mainly focus on the recall of Class 1 (Defect, positive), which related to the miss detection.

5.3 Inspection results

To assess our method, we compare the performance and execution time of our model with other state-of-the-art CNN models including VGG-16, GoogLeNet Inception v1, ResNet-50, and DenseNet-121. The information of each model is in Tables 4 and 5.

We use the focal loss combined with the triplet loss in cosine similarity as our loss function and use the cross-entropy loss as the loss function of other models. VGG has the largest model size because it uses many $3 \times 3 \times 3$ continuous convolutional layers, and these layers have a larger number of kernels, such as 64, 128, 256, and 512. There are 27 parameters in a $3 \times 3 \times 3$ convolution kernel, which are three times more than the parameters of a 3×3 convolution kernel. Therefore, the number of model parameters has significant growth. The model size of our model is the smallest of all models, which is about 0.03 times (=2,408,770/77,702,978) the size of the VGG model.

Table 5 The parameters of our model

Layer_name	Parameters
conv_0	1792
conv_1_1/conv_1	55,328
conv_1_1/conv_2	1040
conv_1_1/conv_3	1040
conv_1_2/conv_1	110,624
conv_1_2/conv_2	2064
conv_1_2/conv_3	2064
tr_conv_1	37,056
conv_2_1/conv_1	165,920
conv_2_1/conv_2	3088
conv_2_1/conv_3	3088
conv_2_2/conv_1	221,216
conv_2_2/conv_2	4112
conv_2_2/conv_3	4112
tr_conv_2	102,720
conv_3_1/conv_1	276,512
conv_3_1/conv_2	5136
conv_3_1/conv_3	5136
conv_3_2/conv_1	331,808
conv_3_2/conv_2	6160
conv_3_2/conv_3	6160
tr_conv_3	201,152
conv_4_1/conv_1	387,104
conv_4_1/conv_2	7184
conv_4_1/conv_3	7184
conv_4_2/conv_1	442,400
conv_4_2/conv_2	8208
conv_4_2/conv_3	8208
fc1	1154

5.3.1 Training stage

In the training stage, we set the learning rate to 0.0001, the batch size to 32, and the number of epochs to 100. Since the volume of the 3D data is quite large ($48 \times 48 \times 48 = 110,592$pixels), so we cannot set a larger batch size at one time. The hyper-parameters of our loss function are determined after several experiments and listed as follows:

Table 4 The information about each model

Model	Depth	#Parameters	Model size(MB)	Loss
VGG-16	16	77,702,978	889	CE Loss
GoogLeNet Inception v1	22	14,427,122	167	CE Loss
ResNet-50	50	46,210,840	530	CE Loss
DenseNet-121	121	11,151,042	131	CE Loss
Our model	14	2,408,770	28.8	Focal loss + Triplet loss

- weighting factor of focal loss $\alpha = 0.75$ (Defect:Normal = 3:1)
- modulating factor of focal loss $\gamma = 2$
- the margin between different classes of triplet loss $m = 0.2$
- scale factor $\lambda = 1$

The learning curves of different models are in Fig. 12a–c, and the training results are in Table 6. After observing the learning curves, we can find that the models converge fast except for the VGG model. The training loss of the VGG model does not decrease, and the accuracy does not change much. The VGG model has no discriminating ability because all the training data are predicted as normal, although the accuracy of the VGG model is 0.91 ($= \frac{315}{30+315}$). This may be caused by data imbalance and a large number of parameters of the model.

We choose the model with the best parameters based on the cross-entropy loss of the validation data. The validation loss curves are in Fig. 12d. The validation results of different models with the best parameters are in Table 7.

5.3.2 Testing stage

In the testing stage, there are two results of our model corresponding to the two parts of our inspection process. We compare the execution time, the CE loss, the precision, and

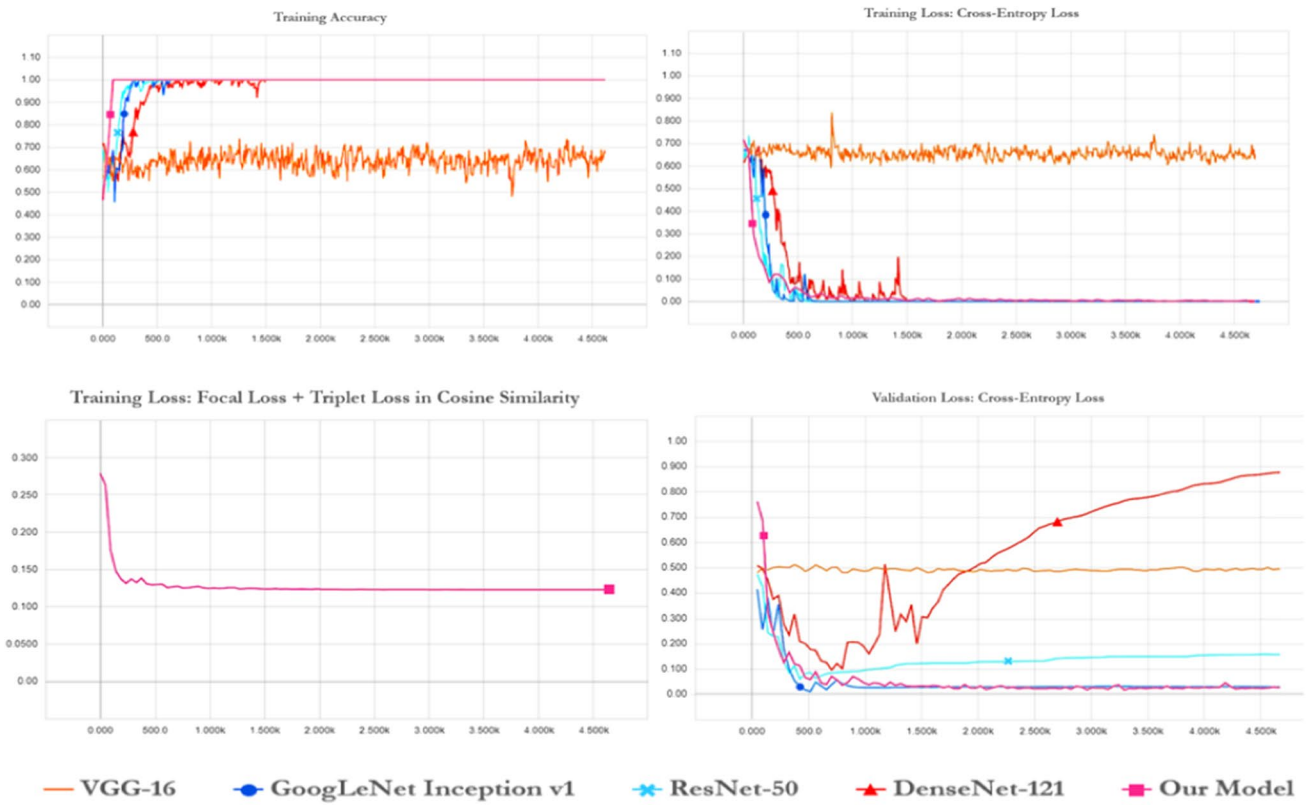

Fig. 12 The learning curves of different models. **a** The training accuracy curves. **b** The training loss curves of the models using cross-entropy loss. Though we do not use CE loss in our model, we also observe the changes in CE loss during the training process. **c** The training loss curve of our model. **d** The validation loss curves of different models

Table 6 The training results of each model

Model	Training time (sec.)	Best epoch/Epoch	Loss	Accuracy
VGG-16	4648.94	16/100	0.470093	0.913943
GoogLeNet Inception v1	493.28	10/100	0.000022	1.0
ResNet-50	779.28	8/100	0.002508	1.0
DenseNet-121	850.92	30/100	0.004144	1.0
Our model	2386.39	99/100	0.123127	1.0

Table 7 The validation results of each model

Model	Best epoch/Epoch	Loss	Accuracy
VGG-16	16/100	0.457370	0.936170 $\left(=\frac{132}{9+132}\right)$
GoogLeNet Inception v1	10/100	0.000710	1.0
ResNet-50	8/100	0.003724	1.0
DenseNet-121	30/100	0.040842	0.978723 $\left(=\frac{8+130}{9+132}\right)$
Our model	99/100	0.020267	0.992908 $\left(=\frac{9+131}{9+132}\right)$

Table 8 The testing time, the model loading time, and the CE loss of each model

Model	Testing time (sec.)	Model loading time (sec.)	Loss
VGG-16	35.945484	1.355382	0.424349
GoogLeNet Inception v1	15.364822	0.573278	0.088349
ResNet-50	23.684031	1.102509	0.042201
DenseNet-121	28.955130	0.698714	0.190918
Our model (CNN)	14.335458	0.282135	0.013444
Our model (CNN + SVM)	21.297390	0.282135	–

the recall of different models. The testing results of different models are in Tables 8, 9, and 10.

The model loading time and the total testing time of our CNN model are the shortest, and the total testing time adding the SVM model is ranked the third, longer than the GoogLeNet model. Our model has the best performance on testing data. Since our model can learn various features via convolutional kernels with different sizes, it can choose the better features to adapt the data to achieve better results. The focal loss can help to solve the data imbalance problem, and the triplet loss can increase the distance between positive and negative samples. Additionally, the results can be further improved after rechecking uncertain data through the second part of our inspection process. GoogLeNet and ResNet also have good testing results. Although DenseNet can connect all the features in the previous layers, its result is not ideal. Maybe the model is overfitting on the training data. VGG is unable to identify the data effectively because it did not learn useful features during the training stage.

Table 9 The number of miss detection and false alarm, the precision, and the recall of each model

Model	#Miss detection	Precision/Recall of class 1	#False alarm	Precision/Recall of class 0
VGG-16	23	–/0	0	0.9867/1.0
GoogLeNet Inception v1	3	0.4082/0.8696	29	0.9982/0.9830
ResNet-50	4	0.5278/0.8261	17	0.9976/0.9900
DenseNet-121	8	0.1546/0.6522	82	0.9951/0.9520
Our Model (CNN)	1	0.7857/0.9565	6	0.9994/0.9965
Our Model (CNN + SVM and not use salt and pepper)	1	0.7333/0.9565	8	0.9994/0.9952
Our model (CNN + SVM)	1	0.8148/0.9565	5	0.9994/0.9971

There are 1708 normal data and 23 defective data in the testing data

Table 10 The number of miss detection and false alarm, the precision, and the recall of our model using different losses.

Our model (using different loss)	#Miss detection	Precision/Recall of class 1	#False alarm	Precision/Recall of class 0
Cross-entropy loss	5	0.3396/0.7826	35	0.9970/0.9790
Focal loss	1	0.3607/0.9565	39	0.9994/0.9766
Triplet loss	1	0.4231/0.9565	30	0.9994/0.9820
Focal loss + Triplet loss	1	0.8148/0.9565	5	0.9994/0.9971

There are 1708 normal data and 23 defective data in the testing data

The slice images of the miss detection data and the false alarm data of different models are shown in Figs. 13 and 14.

5.4 Comparison with SuaKIT

In addition to comparing with several state-of-the-art CNN models, we also compare our model with the results of SuaKIT. Since SuaKIT can only recognize 2D images, we first obtain four-slice images of each 3D solder ball data shown in Fig. 15. Therefore, the number of slice data is four times the original number.

We mainly use the single image classification method. For this method, SuaKIT provides many data augmentation options, four model capacities (small, normal, large, extra-large), and several hyper-parameters, such as epoch and class weight. Finally, we train four SuaKIT models with different hyper-parameters. The information of the SuaKIT models is in Table 11.

Since we will compare the results of each solder ball between our model and the SuaKIT models, we have to integrate the results of four-slice images belonging to the same solder ball. The result of a solder ball is determined as a normal ball only when all four-slice images are predicted as "normal." The results of the SuaKIT models are in Tables 12 and 13.

Although the training time of the SuaKIT model is only about 4 min and much faster than our model (about 40 min), the testing time of our model (28 s.) is faster than the SuaKIT model (240 s.) and has more advantages on the production line. Additionally, our model has the best

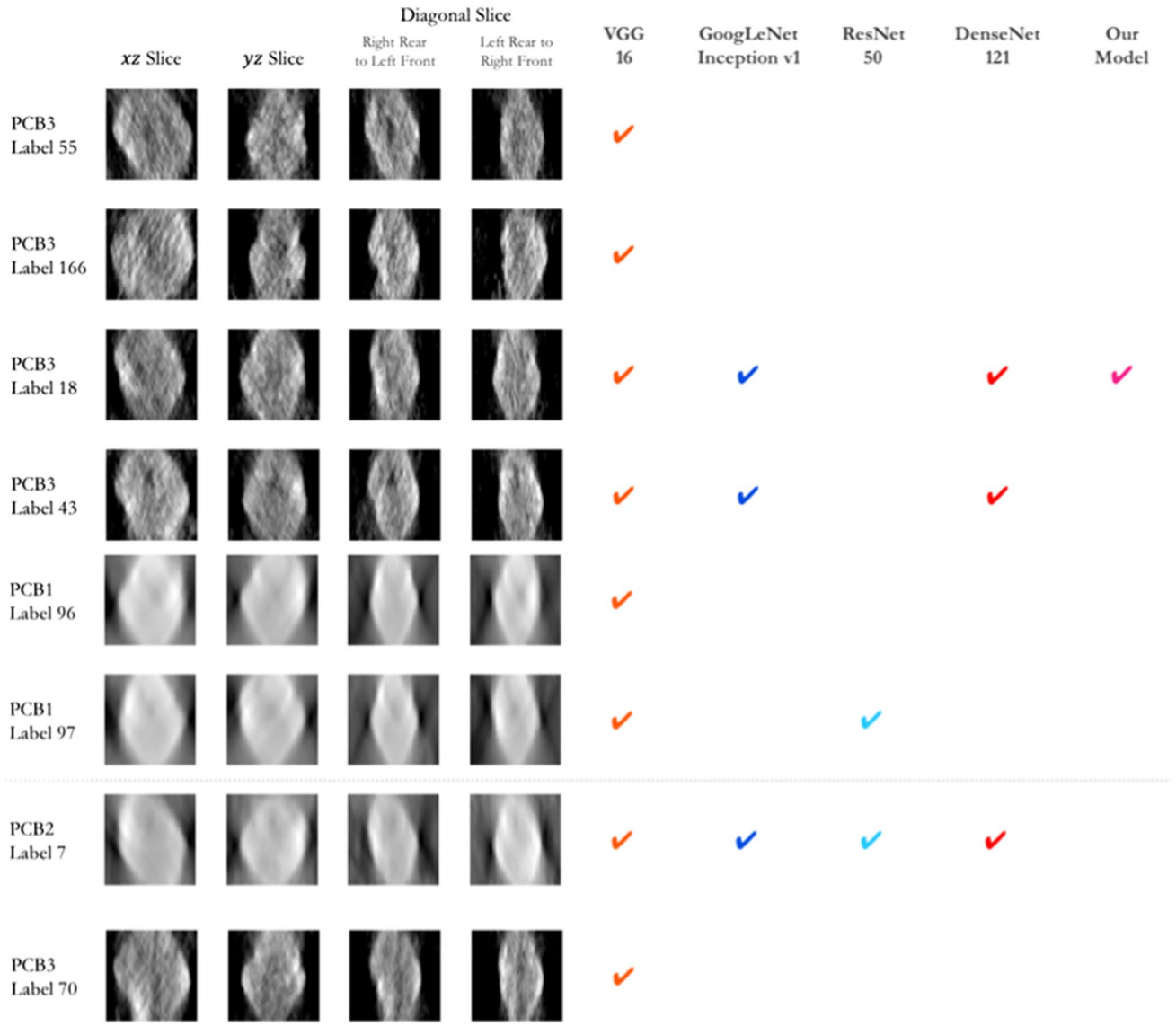

Fig. 13 Part of the slice images of the miss detection data of each model

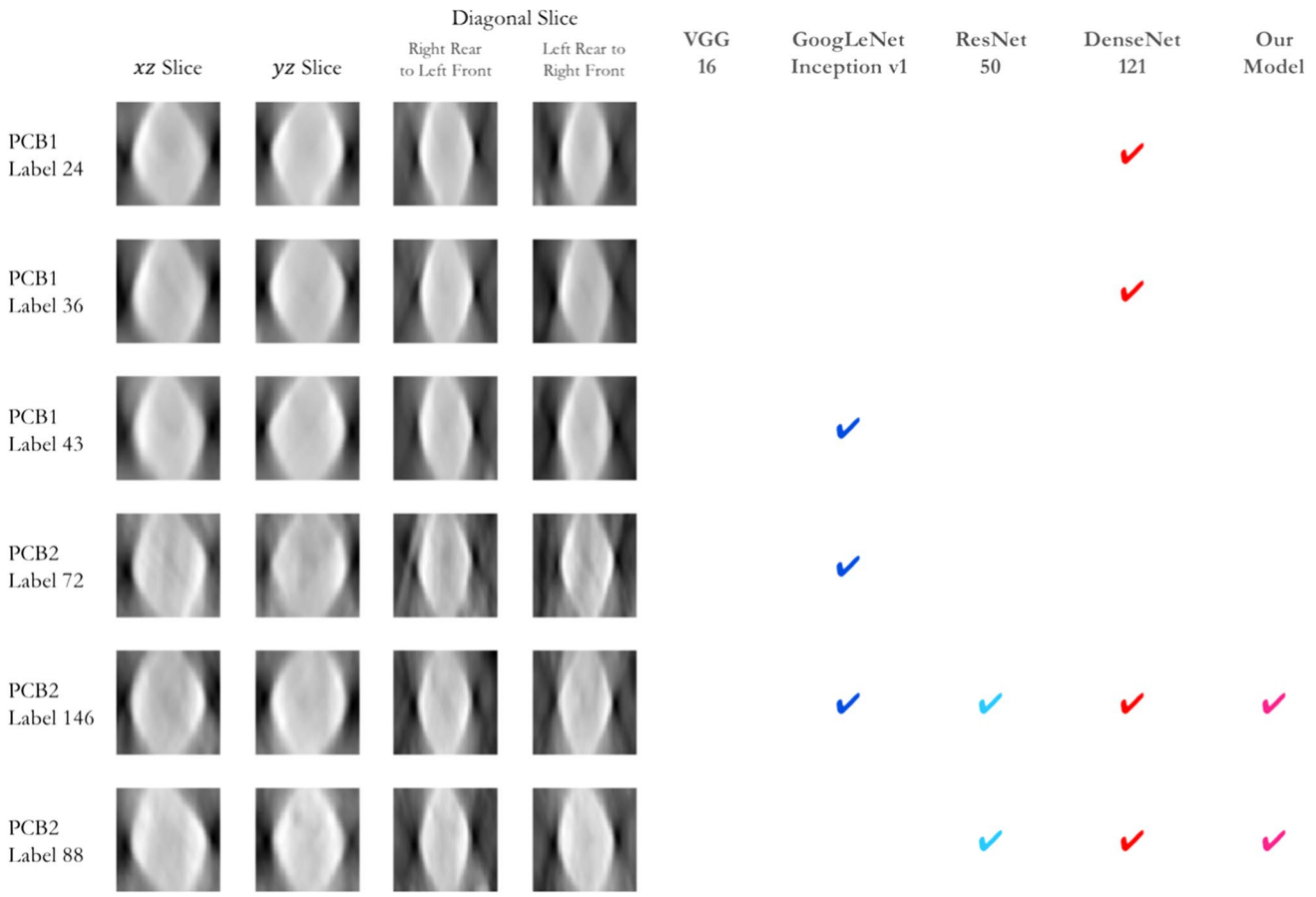

Fig. 14 Part of the slice images of the miss detection data of each model

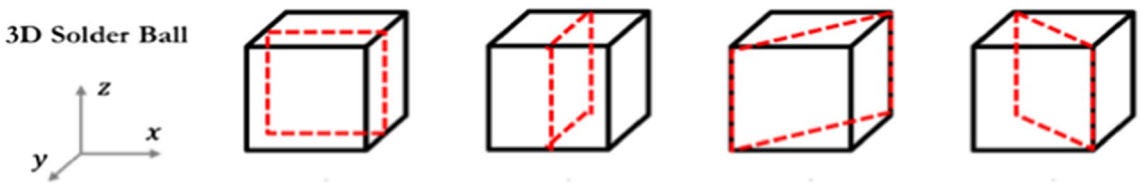

Fig. 15 The slice images of 3D solder ball data

Table 11 The information about four SuaKIT models

Model	Model capacity	Class weight (Normal: Defect)	Augmentation methods
SuaKIT model 1	Small	1:1	Horizontal flip, Vertical flip, Noise, Blur
SuaKIT model 2	Small	1:3	
SuaKIT model 3	Normal	1:1	
SuaKIT model 4	Normal	1:3	

performance on testing data. After observing the testing results of the SuaKIT models, we can find that the models with normal capacity have less miss detection than the models with small capacity, but the number of false alarm is larger than the small one. Through the results of these

Table 12 The testing results of four SuaKIT models and our model

Model	CM (2D Slices)		CM (3D Balls)		Precision/Recall of class 1	Precision/Recall of class 0
SuaKIT Model 1	80	12	16	7	0.3077/0.6957	0.9958/0.9789
	41	6791	36	1672		
SuaKIT Model 2	79	13	16	7	0.5333/0.6957	0.9959/0.9918
	14	6818	14	1694		
SuaKIT Model 3	86	6	19	4	0.2969/0.8261	0.9976/0.9737
	49	6783	45	1663		
SuaKIT Model 4	88	4	19	4	0.1570/0.8261	0.9975/0.9403
	124	6708	102	1606		
Our Model	–		22	**1**	**0.8148/0.9565**	**0.9994/0.9971**
			5	1703		

The number of slice images for testing is 6924 (6832 normal images, 92 defective images)

Table 13 The testing time of four SuaKIT models and our model

Model	Testing time (sec.)
SuaKIT model 1	296
SuaKIT model 2	246
SuaKIT model 3	236
SuaKIT model 4	239
Our model	27.625937

four models, we cannot find the impact of class weight on the model.

6 Conclusion and future works

In this paper, we develop an AI solution combining CNN and SVM algorithms to perform solder ball HIP defect inspection. We use different data augmentation methods to increase the noise robustness of our model and adopt focal loss as well as triplet loss to solve the data imbalance problem caused by rare defective data. The size of our CNN model is the smallest of the CNN models we compare (VGG-16, GoogLeNet Inception v1, ResNet-50, and DenseNet-121), and our inspection method has the best precision of 82% and recall of 96% compared with several classic CNN models and the inspection software SuaKIT. The SuaKIT model 2 has the second-best precision 53%, and the GoogLeNet model has the second-best recall of 87%.

However, our model only accepts fixed-size 3D solder ball data currently. Additionally, there are only three sets of solder ball data with different characteristics. On the production line, the types and sizes of PCBs are ever-changing, and new technologies and products will be developed at any time. Therefore, we will collect more data with various characteristics and sizes for continual learning to increase the generalization of the model and improve the detection rate on the production line.

To solve the data imbalance problem, we will try other strategies. One strategy is using Generative Adversarial Network (GAN) to generate more defective data. Another strategy is One-Class Learning, which uses only normal data (without defective data) to train a model.

Acknowledgements This research was supported by the Ministry of Science and Technology of Taiwan, R.O.C., under Grants MOST 109-2221-E-002-158-MY2 and MOST 108-2221-E-002-140, and by Test Research, Jorgin Technologies, III, Chernger, ARCS Precision Technology, D8AI, PSL, and LVI.

References

1. Wikipedia, Inspection in manufacturing. https://en.wikipedia.org/wiki/Inspection_in_manufacturing, (2019).
2. Encyclopedia, BGA Definition from PC Magazine Encyclopedia, https://www.pcmag.com/encyclopedia/term/38577/bga, 2019.
3. D. Xie, et al. Head in Pillow (HIP) and yield study on SIP and PoP assembly. In: Proceedings of IEEE Electronic Components and Technology Conference, San Diego, California, pp. 752–758, (2009).
4. Gondrom, S., et al.: X-Ray computed laminography: an approach of computed tomography for applications with limited access. Nucl. Eng. Des. **190**(1–2), 141–147 (1999)
5. Agilent Technologies, Chemical analysis, life sciences, and diagnostics-agilent, https://www.agilent.com, 2019.
6. A. Castellanos, et al. Head in pillow X-ray inspection at flextronics, https://smtnet.com/library/files/upload/head-in-pillow-inspection-flextronics.pdf, 2019.
7. Test Research Inc., Test Research Inc., https://www.tri.com.tw/en/index.html, (2019).
8. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995)
9. Deng, L., Yu, D.: Deep learning: methods and applications. Found. Trends® Signal Process. **7**(3–4), 197–387 (2014)
10. Wikipedia, Convolutional neural network. https://en.wikipedia.org/wiki/Convolutional_neural_network, (2019).
11. LeCun, Y., et al.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)

12. K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition. arXiv preprint ,arXiv: 1409.1556, 2014.
13. C. Szegedy, et al. Going deeper with convolutions. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, pp. 1–9, (2015).
14. C. Szegedy, et al. Rethinking the inception architecture for computer vision. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, pp. 2818–2826, (2016).
15. K. He, et al. Deep Residual Learning for Image Recognition. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, pp. 770–778, (2016).
16. G. Huang, et al. Densely connected convolutional networks. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, Hawaii, pp. 4700–4708, (2017).
17. T. C. Tsan, et al. Solder ball 3D reconstruction with X-ray images using filtered back projection. In: TENCON 2018–2018 IEEE Region 10 Conference. IEEE, (2018).
18. Otsu, N.: A threshold selection method from gray-level histograms. IEEE Trans. Syst. Man Cybern. **9**(1), 62–66 (1979)
19. Samet, H., Tamminen, M.: Efficient component labeling of images of arbitrary dimension represented by linear bintrees. IEEE Trans. Pattern Anal. Mach. Intell. **10**(4), 579–586 (1988)
20. T. Y. Lin, et al. "Focal loss for dense object detection. In: Proceedings of IEEE International Conference on Computer Vision. Venice, Italy, pp. 2980–2988, (2017).
21. B. Raj, A Simple Guide to the Versions of the Inception Network. https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202, 2018.
22. F. Schroff, D. Kalenichenko, and J. Philbin, FaceNet: A unified embedding for face recognition and clustering. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, pp. 815–823, (2015).
23. Fawcett, T.: An introduction to ROC analysis. Pattern Recogn. Lett. **27**(8), 861–874 (2006)