Applied Intelligence (2021) 51:1262–1279
https://doi.org/10.1007/s10489-020-01877-z

# A deep learning-based approach for the automated surface inspection of copper clad laminate images

Xiaoqing Zheng[1] · Jie Chen[1] · Hongcheng Wang[1] · Song Zheng[1] · Yaguang Kong[1]

Published online: 19 September 2020


**Abstract**
Surface quality inspection and control are extremely important for electronic manufacturing. The use of machine vision technology to automatically detect the defects of products has become an indispensable means for better quality control. A machine vision-based surface quality inspection system is usually composed of two processes: image acquisition and automatic defect detection. In this paper, we propose a deep learning-based approach for the defect detection of Copper Clad Laminate (CCL) images acquired from an industrial CCL production line. In the proposed approach, a new convolutional neural network (CNN) that realizes fast defect detection while maintaining high accuracy is designed. Our proposed approach makes four contributions. First, we introduce the depthwise separable convolution to reduce the calculation time. Second, we improve the squeeze-and-excitation block to improve network performance. Third, we introduce the squeeze-and-expand mechanism to reduce the computation cost. Fourth, we employ a smoother activation function (Mish) to allow improved information flow. The proposed network is compared with the benchmark CNNs (including Inception, ResNet and MobileNet). The experimental results show that compared with the benchmark networks, our proposed network has achieved the best results regarding the accuracy and suboptimal results in terms of the speed compared with the benchmark networks. Therefore, our proposed method has been integrated into an industrial CCL production line as a guideline for online defective product rejection.

**Keywords** Machine vision · Defect detection · Deep learning · Convolutional neural network · Efficient network

## 1 Introduction

Traditionally, the surface quality inspection of electronic products employs manual detection methods performed by quality inspectors. However, these manual methods have the disadvantages of a low sampling rate, low accuracy, poor real-time performance, low efficiency and high labor intensity, whereas machine vision-based automated surface inspection methods can largely overcome the above drawbacks [1]. At present, the use of machine vision technology, i.e., by adding a vision system to the machine or automatic production line, has become an indispensable technical means of quality control for electronics manufacturing. Machine vision-based surface inspection systems are mainly composed of two processes: image acquisition and defect detection [2]. The image acquisition module consists of an industrial camera, an optical lens, and a light source and its clamping device [1]. The function of the image acquisition module is to collect the surface images of the target products, and the defect detection process refers to the recognition of defects through image processing and classification techniques based on the acquired images.

Traditional machine vision defect detection methods can be divided into four categories: statistical methods, spectral methods, model-based methods, and learning-based methods. To execute these methods, first feature extraction is performed and then defect identification is implemented. The feature extraction methods include the histogram [3], local binary pattern (LBP) [4], and co-occurrence matrix [5] methods in the spatial domain, and the methods of Fourier

This work was supported in part by National Natural Science Foundation of China under grant number U1609212, Zhejiang Provincial Science and Technology Plan under grant number 2019C04021, and Zhejiang Province Public Technology Research Project under grant number LGG20F030002.

✉ Xiaoqing Zheng
zhengxiaoqing@hdu.edu.cn

Jie Chen
1764588497@qq.com

[1] Hangzhou Dianzi University, Hangzhou, China

transform [6], wavelet transform [7] and Gabor transform [8] methods in the transform domain. After the features are extracted, defect recognition is performed by using classifiers such as SVM [9], k-nearest neighbor [10] and random forest [11]. The performance of defect recognition relies to some extent on how well the features are designed and extracted. Therefore, the overall performance depends on how well the manually designed representations can model the properties of the defects, and expertise is the key to the success of these methods, which limits their wide application [12]. In other words, the performance depends strongly on the knowledge, experience, and judgment of the engineers who design the representations of the defects. In recent years, deep learning has achieved very good results in face recognition, speech recognition, and natural language processing. However, this method has been rarely applied to the field of automated surface inspection. The main reasons are as follows: surface defect datasets are normally too small to train deep learning networks, deep learning requires high computing power, and it is laborious to collect and label image samples. Despite these difficulties, as an emerging and promising technology, deep learning has the potential to solve the aforementioned challenges of machine vision-based surface inspection.

Therefore, the research of deep learning-based automated surface inspection has attracted strong attention both from academia and industry. For instance, Chanhee Jang et al. [13] proposed a defect inspection method combining defect probability images and a deep convolutional neural network, which works well on small datasets and removes the human skill requirement. Ruoxu Ren et al. [14] proposed a generic deep learning-based approach which requires small training data for automatic surface inspection. Xiaoqing Zheng et al. [15] presented a generic semi-supervised deep learning approach that requires a small quantity of labeled data for automated surface defect inspection. D. Soukup et al. [16] trained a classical convolutional neural network (CNN) on a database of photometric stereo images of metal surface defects in a purely supervised manner. Je-Kang Park et al. [17] proposed a new surface defect inspection method using CNN and tested several types of deep networks with different depths and layer nodes to select an adequate structure for surface defect inspection. Tian Wang et al. [18] proposed a CNN for defect detection with less prior knowledge regarding the images that is robust to noise. Alessandra Caggiano et al. [19] developed a machine learning approach based on a deep convolutional neural network for on-line fault recognition via automatic image processing to identify material defects.

Among these deep learning-based defect detection approaches, the most frequently applied network is a CNN due to its wide application possibilities in recognizing different patterns [20]. The basic CNN consists of three kinds of layers, namely, the convolutional layer, pooling layer and fully connected layer. The convolutional layer learns the feature representation of the input and generates the feature map output. The pooling layer aims to reduce the computational complexity through a dimensionality reduction of the feature map. The fully connected layer implements the mapping of input data to a one-dimensional feature vector for the final output layer's use or further feature processing. As a CNN has a unique feature learning ability to automatically learn features from image samples, it overcomes the limitation of manual design and exhibits strong reliability. A CNN has a certain degree of invariance to geometric transformation and deformation. Therefore, CNNs have a very bright future in the surface quality inspection area [21].

In our work, we proposed an efficient CNN-based approach for automated surface inspection of copper clad laminate images. The approach has realized the accurate and rapid identification of surface defects of products in CCL high-speed production lines.

## 2 Machine vision-based surface inspection system

### 2.1 System architecture

The machine vision-based automated surface inspection system has the advantages of high precision, high efficiency, high speed and continuous detection, and noncontact measurements. This method mainly consists of two processes: image acquisition and defect detection. The function of the image acquisition process is to collect the surface images of the target products, and the defect detection process refers to the recognition of defects through image processing and detection techniques. Figure 1 shows the architecture and flow-sheet of a machine vision system.

As demonstrated in Fig. 1, the image acquisition process consists of an optical system and an image acquisition unit. The optical system is composed of a smart camera (usually CCD and CMOS), an optical lens and a light source. The optical system generates images that are exported to the image acquisition unit, which transforms image signals into data files that can be processed by computers. The image acquisition unit can control the operations of the camera.

During the defect detection process, the basic image processing unit is applied to implement basic image processing, such as image filtering and feature extraction (geometric features, texture features, projection features). Then, the defect detection and classification unit is used to determine whether an image is a defective image, the severity of the defects and the category of the defects. Finally, the defect detection and classification results are output to an actuator to control and reject the defective products.

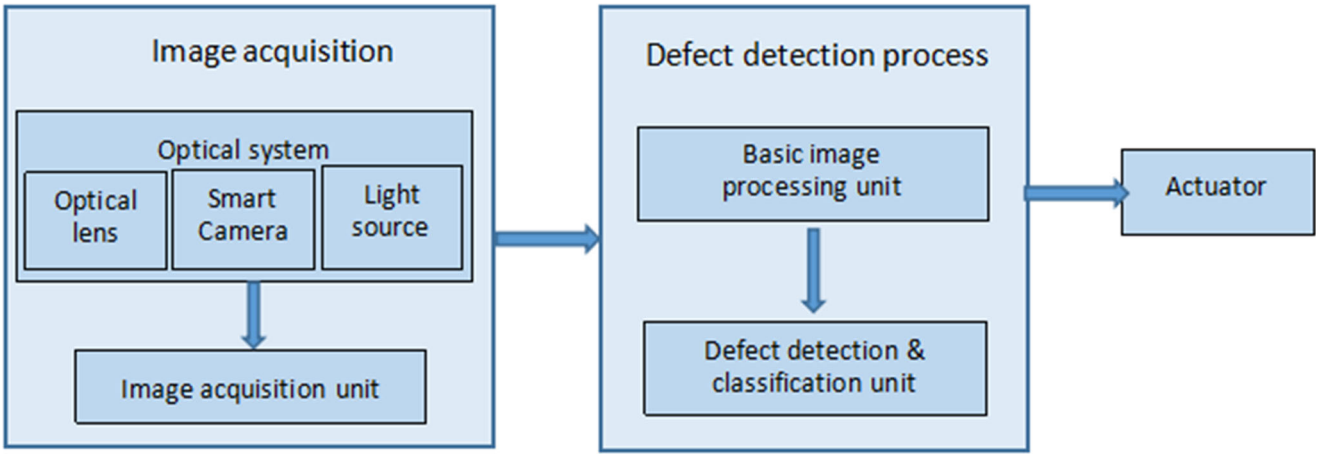



**Fig. 1** Architecture of a machine vision inspection system

In this work, we focus on defect detection and classification unit research by applying classical CNN models, as well as proposing a highly efficient and accurate deep CNN model.

### 2.2 Copper clad laminate surface defects

The image samples we applied in this work are collected from an industrial copper clad laminate (CCL) production line. A CCL is the basic material of the electronics industry and is mainly used to manufacture printed circuit boards. The common surface defects of a CCL include scratches, oil stains, pinholes, inclusions and so on. Our industrial defect dataset divides CCL surface defects into 15 categories, as shown in Fig. 2. Among the fifteen types of defects, most of them are severe defects (types 1, 3, 4, 5, 6, 7, 9, 11, 12, and 15), whereas some of them (types 2, 8, 10, 13, and 14) are nonsevere defects. In addition, type number 16 represents the samples without defects.

## 3 The proposed method

An efficient CNN-based approach for the automated surface inspection of a CCL is proposed. The proposed method aims to achieve a balance between the lowest computation cost and the highest accuracy, as rapid and efficient defect detection is extremely important for the online use of CCL automated surface inspection.

### 3.1 Related work

This section introduces the related algorithms on which our proposed method is based; these algorithms include depthwise separable convolution [22, 23] a squeeze-and-excitation block [24], and a squeeze-and-expand mechanism [25].

#### 3.1.1 Depthwise separable convolution

Depthwise separable convolution consists of a depthwise convolution (channelwise spatial convolution) and a pointwise convolution (1*1 convolution). Specifically, a 3*3 depthwise separable convolution can be decomposed into a 3*3 depthwise convolution, where each input channel is convoluted by applying a 3*3 convolution filter, and a pointwise convolution, which applies a 1*1 convolution on the output of the depthwise convolution to obtain new feature maps. The calculation amount for a standard 3*3 convolution is 3*3*M*N*D*D, where M is the number of input channels, N is the number of output channels, and D*D is the size of the output feature map. For a 3*3 depthwise separable convolution, the calculation amount is:

$$C = 3 * 3 * M * D * D + 1 * 1 * M * N * D * D \quad (1)$$

Therefore, compared with standard 3*3 convolutions, the 3*3 depthwise separable convolution can achieve 8 to 9 times greater computation savings while ensuring almost the same accuracy.

#### 3.1.2 Squeeze-and-excitation block

The squeeze-and-excitation block (SE block) adaptively recalibrates channelwise feature responses between channels and facilitates significant performance improvements [24]. In the SE block, the features are passed through a squeeze operation and an excitation operation. The squeeze operation generates a channel descriptor by aggregating feature maps across their spatial dimensions. The descriptor produces an embedding of the global distribution of channelwise feature responses. The excitation operation takes the embedding as the input and produces a collection of per-channel modulation weights, which are applied to the

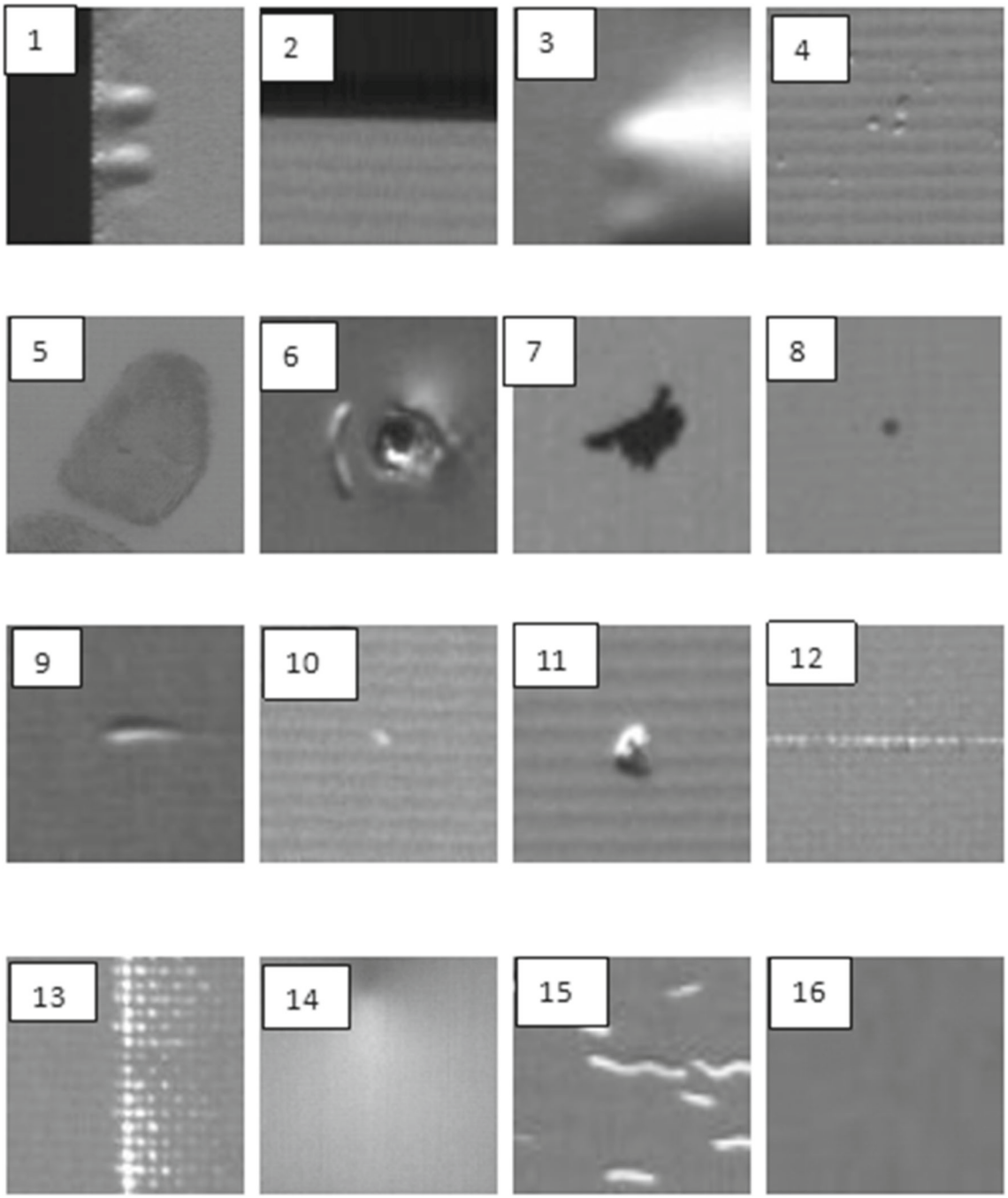



**Fig. 2** Image samples of CCL surface defects ((Type No. 1) Belt G2: severe defect mingled with belt; (Type No. 2) Belt G0: part of the conveyor belt (the dark region); (Type No. 3) Bending G2: severe defects related to material bending; (Type No. 4) Board Mark G2: severe defect caused by multiple pits and dents; (Type No. 5) CST G2: severe defect of corrosion; (Type No. 6) CuBreak G2: severe defect due to broken material (Copper); (Type No. 7) Dark Spot G2: defect of a major dark spot or foreign item; (Type No. 8) DC G0: small dark spot or corrosion; (Type No. 9) PmK G2: severe defect caused by long or multiple pits and dents; (Type No. 10) PnD G0: small pit and dents; (Type No. 11) PnD G2: severe defect caused by a severe pit and dents; (Type No. 12) Scratch G2: severe defect caused by a scratch; (Type No. 13) Shiny Spot G0: minor shiny spot; (Type No. 14) Suction Mark G0: suction mark; and (Type No. 15) Wrinkle G2: severe defect of a wrinkle type)

feature maps to generate the output of the SE block [24]. Finally, the weight of each channel learned is multiplied by the original feature after the activation function to obtain the new feature and improve the training model results.

#### 3.1.3 Squeeze-and-expand mechanism

The squeeze-and-expand mechanism reduces the quantity of parameters by replacing the 3*3 convolution kernel with

1*1 convolution kernels and by decreasing the number of input channels of the 3*3 convolution kernel. This mechanism mainly includes two layers of convolution operations: one is the squeeze layer with a 1*1 convolution kernel, and the other is the expand layer with two branches, each using 1*1 and 3*3 convolution kernels. The number of filters in the squeeze layer is less than the total number of filters in the expand layer. Finally, the two branches of the expand layer are concatenated, and the final feature map is output.

## 3.2 The proposed method

We propose an efficient convolutional neural network for defect detection by first introducing a depthwise separable convolution, a squeeze-and-expand mechanism and an improved squeeze-and-excitation block and then combining them in a neat and orderly fashion to form a compact and efficient network structure. The proposed method aims to achieve high accuracy at a high computation speed.

The main contributions of our proposed method are as follows:

1. Introduce a depthwise separable convolution to reduce the number of model parameters and calculations.

2. Improve the squeeze-and-excitation block. In this block, the importance of each feature channel can be obtained automatically by learning. Then, based on the obtained information, the useful features can be enhanced, and the features that are not useful for the detection task can be suppressed.

3. Introduce a squeeze-and-expand mechanism to significantly reduce the number of parameters and improve the accuracy when the number of parameters is limited.

4. Improved activation function. A smoother activation function (Mish) is employed as the activation function to allow a deeper flow and better spread of information.

### 3.2.1 Proposed CNN

To obtain the highest accuracy with the fewest model parameters and calculations, we designed two building blocks referred to as depthwiseFire and depthwiseResidual in the proposed CNN. These building blocks are illustrated in Fig. 3. The structure of the DepthwiseFire block is inspired by the squeeze-and-expand mechanism, which has two branches. This structure helps to achieve the highest accuracy with the lowest computation cost. The DepthwiseResidual block has a parameter-free shortcut connection added, which is inspired by the residual structure [26]. The shortcut connection helps the network to converge, which helps us to find the appropriate network depth through experiments of stacking different numbers of layers together. Furthermore, depthwise separable convolutions and queeze-and-excitation blocks are employed by both blocks to improve network performance further.

The structure of the DepthwiseFire block can reduce the parameters and calculation amount by using the squeeze-and-expand mechanism, which is illustrated in Fig. 3a DepthwiseFire. A squeeze layer is used to reduce the number of input channels to the expander layer with 3*3 convolutions. An expanding layer has two branches. One branch uses 1*1 convolutions instead of 3*3 convolutions to reduce the calculation amount. The other branches use 3*3 depthwise separable convolutions with a reduced number of input channels, which can achieve good accuracy with fewer calculation parameters. Furthermore, the two-branch structure with 3*3 convolutions and an improved SE block helps to greatly improve the detection accuracy. Specifically, as illustrated in Fig. 3, the block DepthwiseFire starts with a squeeze layer by performing 1*1 convolutions, followed by an expand layer with two branches. The left branch performs a 1*1 convolution, and the right branch performs a 3*3 depthwise separable convolution, followed by SE Block and 1*1 convolutions. Then, the outputs of the two branches are concatenated together. Here, we use a 1*1 convolution rather than a 3*3 convolution in the squeeze layer, through which the number of parameters of a convolution operation is reduced by 9 times. The relationship between the number of channels (E) in each branch of the expand layer and the number of channels (S) in the squeeze layer is as follows:

$$E = 4 * S \tag{2}$$

In this way, the input channels of the expand layer are decreased by a factor of four, resulting in a significant reduction in the calculation amount. In addition, the use of the depthwise separable convolution can save 8 to 9 times the amount of calculation while ensuring almost the same accuracy. Therefore, the combination of the squeeze-expand mechanism and depthwise separable convolution in DepthwiseFire results in a significant reduction in the number of calculations and greatly improves the calculation efficiency.

The structure of the DepthwiseResidual block reduces the parameters and computation by using the depthwise separable convolution operation. In addition, the residual structure and SE block improve the performance of the model. Specifically, in the DepthwiseResidual block, two 3*3 depthwise separable convolution layers and an improved SE block are stacked together with a parameter-free identity shortcut connection added. By adding the SE block, processing is added between two adjacent layers, which makes the information interaction between channels possible and further improves the accuracy of the network. In addition, the shortcut connection helps the network to converge, and depthwise separable convolution helps to save computation cost

The details of the improved squeeze-and-excitation block (SE block) are illustrated in Fig. 4; it is mainly divided into

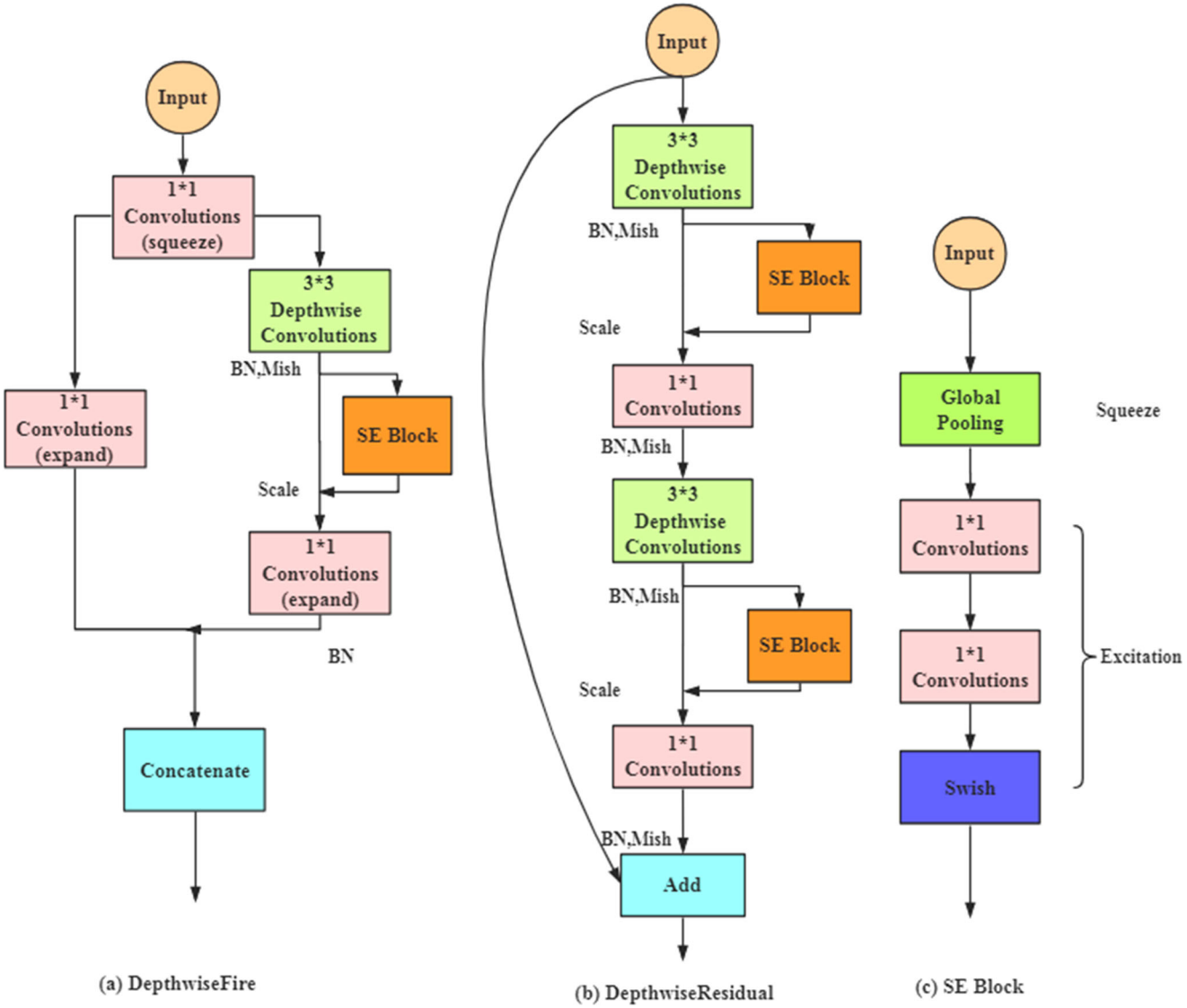



**Fig. 3** The building blocks: **a** DepthwiseFire, **b** DepthwiseResidual, and **c** SE Block

three steps: first, the squeeze operation, then the excitation operation, and finally, the scale operation. In the improved SE block, the size of the input feature map is H*W*C, where H is the height, W is the width, and C is the number of channels. The feature map first passes through an average pooling layer to obtain a feature map of size 1*1*C (which can be regarded as the squeeze operation). Thus, the two-dimensional feature channel is changed to one dimension. This one-dimensional feature has a global receptive field, and its output dimension equals the channel number of the input feature map. This squeeze operation characterizes the global distribution of the responses on the feature channel and makes the one-dimensional features obtain the previous global view of H*W and a wider receptive field. The next step is the operation of excitation. To reduce the complexity of the model and improve the generalization ability, the bottleneck structure containing two convolution layers is used in the excitation operation. The input feature map is passed through a 1*1 convolution layer to generate an output feature map of size 1*1*C/16. This 1*1 convolution layer plays a role of channel reduction. Then, the feature map is input into the other 1*1 convolutional layer to generate a feature map of size 1*1*C. This 1*1 convolutional layer is aimed to restore the original channel. The purpose of the two-layer excitation operation is to reduce the number of channels and thus the amount of

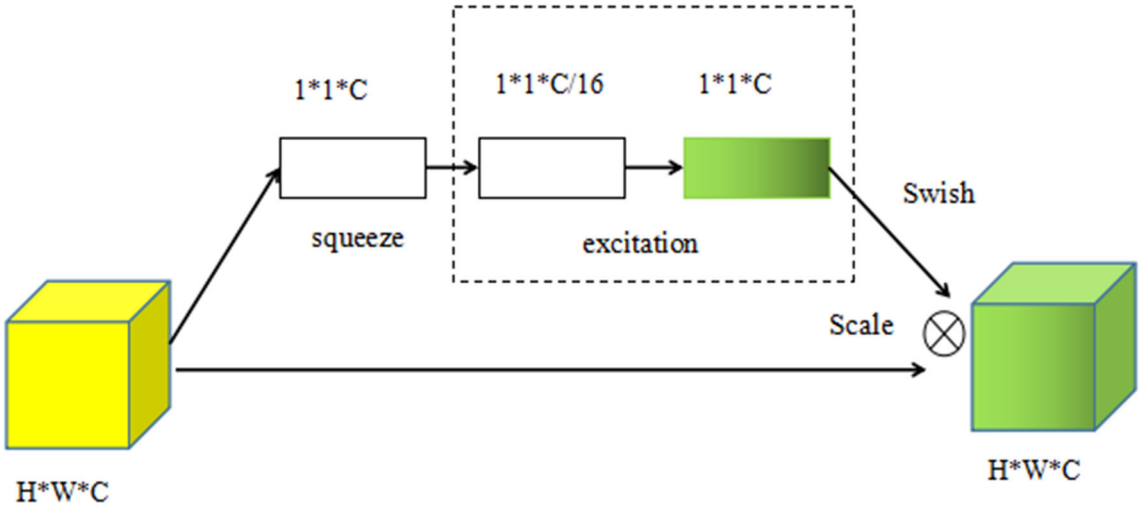



**Fig. 4** The structure of the Squeeze-and-Excitation Block (SE Block)

computation. Then, a swish activation layer is applied, and the output feature map of size 1*1*C is obtained. The formula of swish is as follows:

$$f(x) = x * sigmod(x) \tag{3}$$

In this operation, we improve the original SE block by using a 1*1 convolution to replace its original fully connected layer.

The final step is a scale operation. We regard the activation value of the excitation operation to be a representation of the importance of each feature channel after feature selection and then multiply the activation value by their previous features to complete the channel dimension matching and recalibration of original features. Therefore, the training model can achieve better accuracy.

Furthermore, we use Mish as the activation function instead of Relu in our proposed network. In large-scale neural networks, with an increase in the layer depth, the accuracy of activation functions such as ReLU decreases rapidly, whereas Mish [27] is able to maintain accuracy and propagate information correctly. The formula of Mish is as follows:

$$f(x) = x * tanh(ln(1 + e^x)) \tag{4}$$

Based on these two building blocks (depthwiseFire and depthwiseResidual), we propose an efficient convolutional neural network by stacking them together. The overall structure of our proposed network is shown in Table 1. To reduce the overfitting of the model, dropout and batch normalization [33] are also used before each convolution operation.

The network structure and process are described as follows: The size of the input picture is 224*224.

A 3*3 convolution with 32 filters and stride 2 results in a 112*112*32 output.

A maxpool layer with a kernel size of 3*3 and stride 2 results in a 56*56*32 output.

A DepthwiseFire, resulting in a 56*56*32 output.

A DepthwiseFire, resulting in a 56*56*64 output.

A DepthwiseFire, stride 2, resulting in a 28*28*96 output.

A maxpool layer with a kernel size of 3*3 and stride 2 results in a 14*14*128 output.

A DepthwiseFire, resulting in a 14*14*256 output.

A maxpool layer with a 3*3 filter size and stride 2 results in a 7*7*256 output.

Two DepthwiseResidual, resulting in a 7*7*256 output.

Two DepthwiseResidual, resulting in a 7*7*512 output.

A DepthwiseResidual, resulting in a 7*7*1024 output.

An average pool with a kernel size of 7*7, resulting in a 1*1*1024 output.

A fully connected layer and classifier to accomplish classification.

### 3.2.2 Defect detection architecture

The defect detection architecture based on our proposed CNN is demonstrated in Fig. 5. The detection process is mainly divided into two parts: the training stage and the detection stage. In the training stage, image acquisition, image preprocessing and data augmentation are carried out first. Then, the images are input into our proposed network for iterative training. During iteration, the model parameters and structures are adjusted to obtain a well-trained model that meets the high accuracy requirement of CCL defect detection. In the detection phase, we collect the target images to be detected and input them into the well-trained

**Table 1** Structure of our proposed network

| Group name | Output size | Filter shape | Stride |
|---|---|---|---|
| conv1 | 112*112*32 | 3*3,32 | 2 |
| pool | 56*56*32 | 3*3 Maxpool | 2 |
| DepthwiseFire | 56*56*32 | DepthwiseFire, 32 | 1 |
| DepthwiseFire | 56*56*64 | DepthwiseFire, 64 | 1 |
| DepthwiseFire | 28*28*96 | DepthwiseFire, 96 | 2 |
| DepthwiseFire | 28*28*128 | DepthwiseFire, 128 | 1 |
| Pool | 14*14*128 | 3*3 Maxpool | 2 |
| DepthwiseFire | 14*14*192 | DepthwiseFire, 192 | 1 |
| DepthwiseFire | 14*14*256 | DepthwiseFire, 256 | 1 |
| Pool | 7*7*256 | 3*3 Maxpool | 2 |
| DepthwiseResidual | 7*7*256 | DepthwiseResidual, 256 | 1 |
| DepthwiseResidual | 7*7*256 | DepthwiseResidual, 256 | 1 |
| DepthwiseResidual | 7*7*512 | DepthwiseResidual, 512 | 1 |
| DepthwiseResidual | 7*7*512 | DepthwiseResidual, 512 | 1 |
| DepthwiseResidual | 7*7*1024 | DepthwiseResidual, 1024 | 1 |
| Pool | 1*1*1024 | 7*7 Average pool | 1 |
| Dense | 1*1*15 | Dense | 1 |

model to complete defect detection and obtain the detection results.

## 4 Experiments and discussions

The experiments are carried out under the Linux operating system in a server configured Tesla V100 GPU with 32G memory. The construction, training and testing of neural networks are realized by calling the Keras deep learning library in Python language.

### 4.1 Experimental methods

In our work, the state of the art efficient CNN architectures including Inception [28, 29], ResNet [26] and MobileNet [22, 23], are employed as benchmark networks. We compare the experimental results of our proposed network with those of the benchmark networks. Inception and ResNet are large-sized CNN models that have achieved outstanding performance on ImageNet [30]. Inception-v3 factorizes convolutions into smaller convolutions or asymmetric convolutions to reduce the computation cost. For instance, the 5*5 convolution is decomposed into two 3*3 convolution operations, and the convolution with a kernel size of n*n is decomposed into two convolutions with sizes of 1*n and n*1. At the same time, its filter bank sizes are extended to maintain the computation ability. ResNet utilizes efficient residual networks to address the degradation problem; that is, the training accuracy saturates and then degrades rapidly as the number of network layers increases. As the degradation problem is addressed, the network performance can be improved by simply increasing the network depth. In ResNet-50, a bottleneck building block is applied to improve efficiency, where 1*1, 3*3, and 1*1 convolutions are stacked together with a shortcut connection, where the first 1*1 convolution is responsible for dimensionality reduction, and the last 1*1 convolution is for dimension restoration. MobileNet is a lightweight network designed for mobile and embedded vision applications. This algorithm utilizes depthwise separable convolution to factorize a standard convolution into a depthwise convolution and pointwise convolution to greatly reduce the computation and model size. Inception, ResNet and MobileNet are all efficient networks that aim to reduce computation complexity while retaining good accuracy. We choose them as benchmark methods, as rapid and efficient defect detection is extremely important for the online use of machine vision-based CCL surface inspection.

#### 4.1.1 Data augmentation

Data augmentation is an effective method to address the requirement of deep learning for large-scale datasets. This method can effectively increase available datasets, reduce overfitting, enhance model generalization and improve training accuracy. Transfer learning [31, 32] or training from scratch can benefit from data augmentation. Data augmentation is implemented by performing an image transformation, through which artificial data are created based on the original datasets [32]. Our image samples are collected from an actual production line and cannot meet the

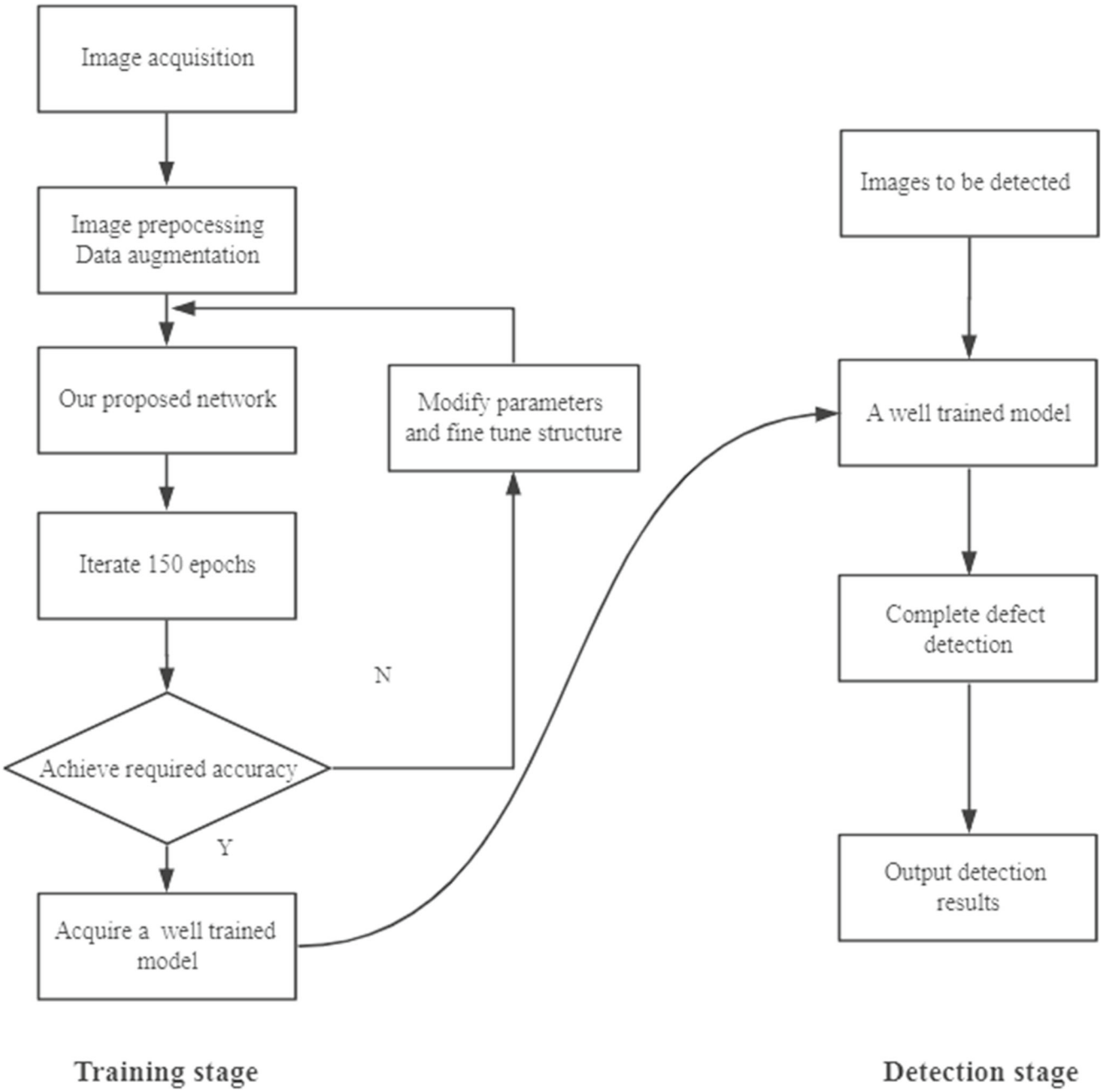



**Fig. 5** The defect detection architecture based on our proposed network

big data needs of deep learning. Therefore, we artificially enlarge our dataset by label-preserving transformations. We test the common transformation methods, including flipping, random cropping, rescaling and color shifting, and find that flipping and noise reduction are best for the performance improvement of CCL defect detection. Therefore, the methods of flipping and noise reduction are employed in our work to expand the original CCL dataset to address the problem of insufficient samples. Several image samples after data augmentation are illustrated in Fig. 6.

Specifically, the 12390 defect image samples collected from an industrial CCL production line were investigated and manually sorted and labeled as 15 categories. This work is time-consuming and laborious, but it is worthwhile and necessary to achieve good image classification results in the upcoming experiments. Then, data augmentation methods

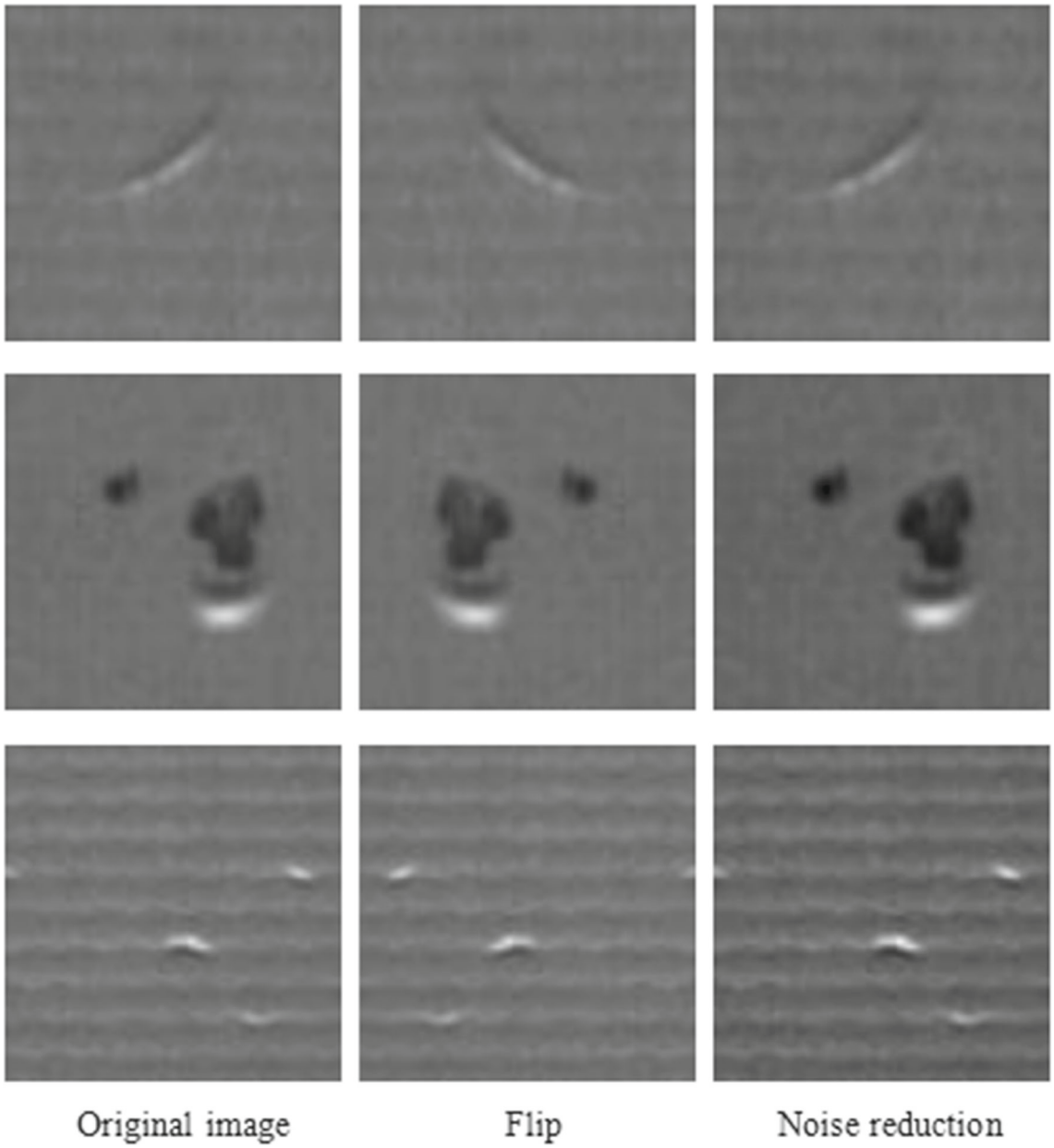



**Fig. 6** Image samples after data augmentation

of flipping and noise reduction were applied to expand the labeled dataset. We first flipped all of the original images and then denoised the original images as well as the flipped images. In this way, we expanded the original dataset by four times. The original dataset consists of 12390 product defect images labeled as 15 categories, while the expanded dataset contains 49560 images also labeled as 15 categories.

As shown in Table 2, we divided the expanded dataset into a training set, a validation set and a testing set at a ratio of 8:1:1, resulting in three sets with 36768, 4956, and 4956 image samples, respectively. We used this training, validation, and testing set for all of the deep learning networks' training carried out in this work.

#### 4.1.2 Network training

The networks including MobileNet-v2, Inception-v3, Resnet-50 and our proposed network were trained to classify CCL defect images. For MobileNet-v2, Inception-v3, and Resnet-50, we only utilized their model structures and trained them from scratch using randomly initialized weights. Several common parameters were set for training as follows.

**Table 2** Data quantity and distribution

| Type | Total | Train | Valid | Test |
|---|---|---|---|---|
| | 49560 | 36768 | 4956 | 4956 |
| Type No. 1 | 4968 | 3974 | 497 | 497 |
| Type No. 2 | 1236 | 988 | 124 | 124 |
| Type No. 3 | 1804 | 1444 | 180 | 180 |
| Type No. 4 | 5120 | 4096 | 512 | 512 |
| Type No. 5 | 4432 | 3546 | 443 | 443 |
| Type No. 6 | 2672 | 2138 | 267 | 267 |
| Type No. 7 | 1592 | 1274 | 159 | 159 |
| Type No. 8 | 2796 | 2236 | 280 | 280 |
| Type No. 9 | 4588 | 3670 | 459 | 459 |
| Type No. 10 | 7448 | 5958 | 745 | 745 |
| Type No. 11 | 2372 | 1898 | 237 | 237 |
| Type No. 12 | 2852 | 2282 | 285 | 285 |
| Type No. 13 | 5044 | 4036 | 504 | 504 |
| Type No. 14 | 308 | 246 | 31 | 31 |
| Type No. 15 | 2328 | 1862 | 233 | 233 |

Set the batch size to 100, which means extracting 100 samples from the training set each time to participate in training.

Use the ADMA optimizer (beta 1 = 0.9, beta 2 = 0.999)

Apply the cross-entropy loss function.

$$Loss = -[y1 * logy2 + (1 - y1) * log(1 - y2)] \quad (5)$$

where y1 and y2 denote the original label and the recognized label of the samples,respectively.

Set the Epoch number to 150. The accuracy is evaluated at the end of each epoch. If the accuracy was not improved within 20 epochs during training, then the training was terminated in advance.

The training progress of the four networks is illustrated in Fig. 7. As shown in Fig. 6, it does not take many iteration epochs for the four networks to converge, and our proposed network achieves the best training accuracy. The convergence curves of our proposed method are also illustrated in Fig. 8. The convergence curve of the validation set fits very well to that of the corresponding training set, demonstrating that overfitting does not occur with our proposed method.

## 4.2 Results and discussions

The performance of MobileNet-v2, Inception-v3, ResNet-50 and our proposed network on the CCL testing set with 4956 image samples is demonstrated in Table 3. The overall accuracy and the accuracy of each type of CCL surface

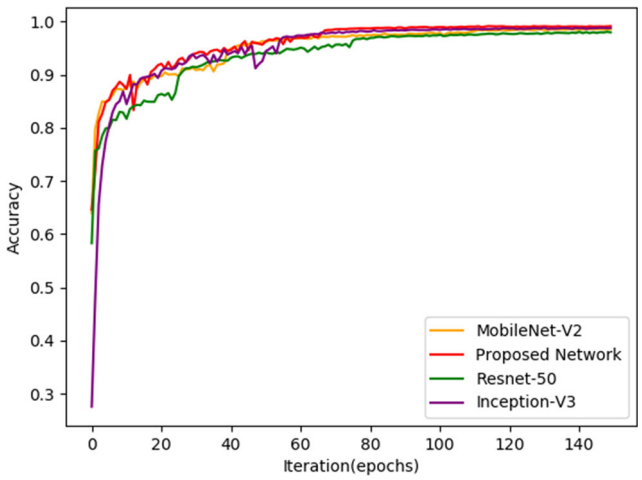



**Fig. 7** Accuracy of the four networks for the CCL dataset during training iterations

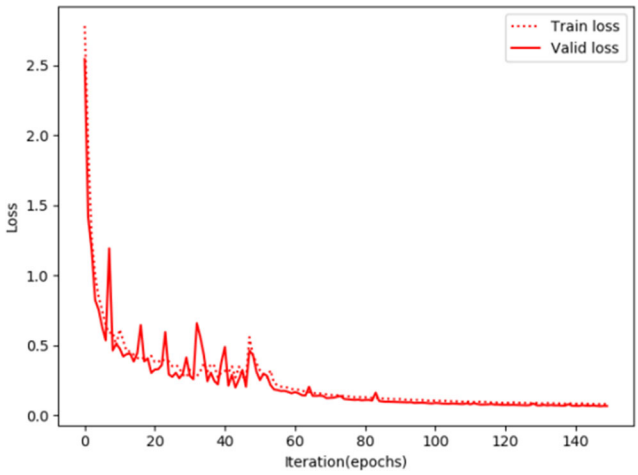



**Fig. 8** Convergence curves of our proposed method during training and validation iterations

**Table 3** Performance of four networks

| Accuracy | MobileNet-v2 | Inception-v3 | Resnet-50 | Proposed network |
|---|---|---|---|---|
| Overall | 98.93% | 99.07% | 98.53% | **99.15**% |
| Type No. 1 | 100.00% | 100.00% | 100.00% | 100.00% |
| Type No. 2 | 100.00% | 100.00% | 100.00% | 100.00% |
| Type No. 3 | 100.00% | 100.00% | 98.33% | 100.00% |
| Type No. 4 | 97.85% | 99.02% | 96.29% | 98.63% |
| Type No. 5 | 99.55% | 99.77% | 98.19% | 99.77% |
| Type No. 6 | 95.51% | 98.88% | 97.38% | 96.25% |
| Type No. 7 | 98.74% | 91.82% | 96.86% | 95.60% |
| Type No. 8 | 97.86% | 98.21% | 95.36% | 98.57% |
| Type No. 9 | 98.47% | 97.82% | 98.69% | 99.35% |
| Type No. 10 | 99.46% | 99.60% | 99.73% | 99.33% |
| Type No. 11 | 96.62% | 97.89% | 98.31% | 97.89% |
| Type No. 12 | 100.00% | 100.00% | 99.65% | 100.00% |
| Type No. 13 | 99.80% | 100.00% | 99.40% | 100.00% |
| Type No. 14 | 100.00% | 100.00% | 100.00% | 100.00% |
| Type No. 15 | 100.00% | 99.57% | 99.14% | 100.00% |

defect (depicted in Fig. 2) are listed. The accuracy of each defect type equals the number of correctly classified images divided by the total number of images of this type.

Furthermore, the confusion matrix results are illustrated in Figs. 9, 10, 11 and 12. In a confusion matrix, the first column contains the names of the true classes, and the first row corresponds to the names of the predicted classes. The diagonal cells correspond to items that are classified correctly.

We can conclude from Table 3 and Figs. 9, 10, 11 and 12 that our proposed method achieves the highest accuracy compared with the three benchmark networks and meets

| | N0.1 | N0.2 | N0.3 | N0.4 | N0.5 | N0.6 | N0.7 | N0.8 | N0.9 | N0.10 | N0.11 | N0.12 | N0.13 | N0.14 | N0.15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| N0.1 | 497 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N0.2 | 0 | 124 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N0.3 | 0 | 0 | 180 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N0.4 | 0 | 0 | 0 | 501 | 0 | 0 | 0 | 0 | 2 | 9 | 0 | 0 | 0 | 0 | 0 |
| N0.5 | 0 | 0 | 0 | 0 | 441 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N0.6 | 0 | 0 | 0 | 0 | 0 | 255 | 0 | 0 | 0 | 1 | 11 | 0 | 0 | 0 | 0 |
| N0.7 | 0 | 0 | 0 | 0 | 0 | 0 | 157 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N0.8 | 0 | 0 | 0 | 0 | 0 | 0 | 6 | 274 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N0.9 | 0 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 452 | 0 | 0 | 0 | 0 | 0 | 1 |
| N0.10 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 1 | 741 | 0 | 0 | 0 | 0 | 0 |
| N0.11 | 0 | 0 | 0 | 2 | 0 | 5 | 0 | 0 | 0 | 1 | 229 | 0 | 0 | 0 | 0 |
| N0.12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 285 | 0 | 0 | 0 |
| N0.13 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 503 | 0 | 0 |
| N0.14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 31 | 0 |
| N0.15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 233 |

**MobileNet-v2 (98.93%)**

**Fig. 9** Confusion matrix of MobileNet-v2

| | N0.1 | N0.2 | N0.3 | N0.4 | N0.5 | N0.6 | N0.7 | N0.8 | N0.9 | N0.10 | N0.11 | N0.12 | N0.13 | N0.14 | N0.15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| N0.1 | 497 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N0.2 | 0 | 124 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N0.3 | 0 | 0 | 180 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N0.4 | 0 | 0 | 0 | 505 | 0 | 0 | 0 | 0 | 2 | 4 | 1 | 0 | 0 | 0 | 0 |
| N0.5 | 0 | 0 | 0 | 0 | 442 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N0.6 | 0 | 0 | 0 | 0 | 0 | 257 | 0 | 0 | 0 | 2 | 8 | 0 | 0 | 0 | 0 |
| N0.7 | 0 | 0 | 0 | 0 | 0 | 0 | 152 | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N0.8 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 276 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| N0.9 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 456 | 0 | 0 | 0 | 0 | 0 | 0 |
| N0.10 | 0 | 0 | 0 | 4 | 0 | 1 | 0 | 0 | 0 | 740 | 0 | 0 | 0 | 0 | 0 |
| N0.11 | 0 | 0 | 0 | 1 | 0 | 4 | 0 | 0 | 0 | 0 | 232 | 0 | 0 | 0 | 0 |
| N0.12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 285 | 0 | 0 | 0 |
| N0.13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 504 | 0 | 0 |
| N0.14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 31 | 0 |
| N0.15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 233 |

**Proposed network (99.15%)**

**Fig. 10** Confusion matrix of Inception-v3

the high-precision defect detection requirement of a CCL production line. Another observation is that there are several types of defects that are easily misclassified by all four networks. The six types with the highest classification errors are illustrated in Table 4. These defect types are listed in descending order from top 1 to top 6.

As shown in Table 4, the six types of CCL defects that are most easily misclassified are defect type No. 4, type

| | N0.1 | N0.2 | N0.3 | N0.4 | N0.5 | N0.6 | N0.7 | N0.8 | N0.9 | N0.10 | N0.11 | N0.12 | N0.13 | N0.14 | N0.15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| N0.1 | 497 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N0.2 | 0 | 124 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N0.3 | 0 | 3 | 177 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N0.4 | 0 | 0 | 0 | 493 | 0 | 0 | 0 | 0 | 8 | 10 | 0 | 0 | 0 | 0 | 1 |
| N0.5 | 0 | 1 | 0 | 0 | 435 | 0 | 5 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N0.6 | 0 | 1 | 0 | 0 | 0 | 260 | 0 | 0 | 0 | 2 | 4 | 0 | 0 | 0 | 0 |
| N0.7 | 0 | 0 | 0 | 0 | 0 | 0 | 154 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N0.8 | 0 | 0 | 0 | 0 | 0 | 0 | 13 | 267 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N0.9 | 0 | 2 | 0 | 3 | 0 | 0 | 0 | 0 | 453 | 0 | 1 | 0 | 0 | 0 | 0 |
| N0.10 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 743 | 0 | 0 | 0 | 0 | 0 |
| N0.11 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 2 | 233 | 0 | 0 | 0 | 0 |
| N0.12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 284 | 0 | 0 | 1 |
| N0.13 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 501 | 0 | 0 |
| N0.14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 31 | 0 |
| N0.15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 231 |

**ResNet-50 (98.53%)**

**Fig. 11** Confusion matrix of ResNet-50

| | N0.1 | N0.2 | N0.3 | N0.4 | N0.5 | N0.6 | N0.7 | N0.8 | N0.9 | N0.10 | N0.11 | N0.12 | N0.13 | N0.14 | N0.15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| N0.1 | 497 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N0.2 | 0 | 124 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N0.3 | 0 | 0 | 180 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N0.4 | 0 | 0 | 0 | 505 | 0 | 0 | 0 | 0 | 2 | 4 | 1 | 0 | 0 | 0 | 0 |
| N0.5 | 0 | 0 | 0 | 0 | 442 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N0.6 | 0 | 0 | 0 | 0 | 0 | 257 | 0 | 0 | 0 | 2 | 8 | 0 | 0 | 0 | 0 |
| N0.7 | 0 | 0 | 0 | 0 | 0 | 0 | 152 | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N0.8 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 276 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| N0.9 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 456 | 0 | 0 | 0 | 0 | 0 | 0 |
| N0.10 | 0 | 0 | 0 | 4 | 0 | 1 | 0 | 0 | 0 | 740 | 0 | 0 | 0 | 0 | 0 |
| N0.11 | 0 | 0 | 0 | 1 | 0 | 4 | 0 | 0 | 0 | 0 | 232 | 0 | 0 | 0 | 0 |
| N0.12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 285 | 0 | 0 | 0 |
| N0.13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 504 | 0 | 0 |
| N0.14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 31 | 0 |
| N0.15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 233 |

**Proposed network (99.15%)**

**Fig. 12** Confusion matrix of proposed network

**Table 4** The six defect types with top error rates for the four networks

| Models | Error Rate Top 1 | Top 2 | Top 3 | Top 4 | Top 5 | Top 6 |
|---|---|---|---|---|---|---|
| MobileNet-v2 | Type No. 6 | Type No. 11 | Type No. 4 | Type No. 8 | Type No. 9 | Type No. 7 |
| Inception-v3 | Type No. 7 | Type No. 9 | Type No. 11 | Type No. 8 | Type No. 6 | Type No. 4 |
| ResNet-50 | Type No. 8 | Type No. 4 | Type No. 7 | Type No. 6 | Type No. 11 | Type No. 3 |
| Proposed network | Type No. 7 | Type No. 6 | Type No. 11 | Type No. 8 | Type No. 4 | Type No. 10 |

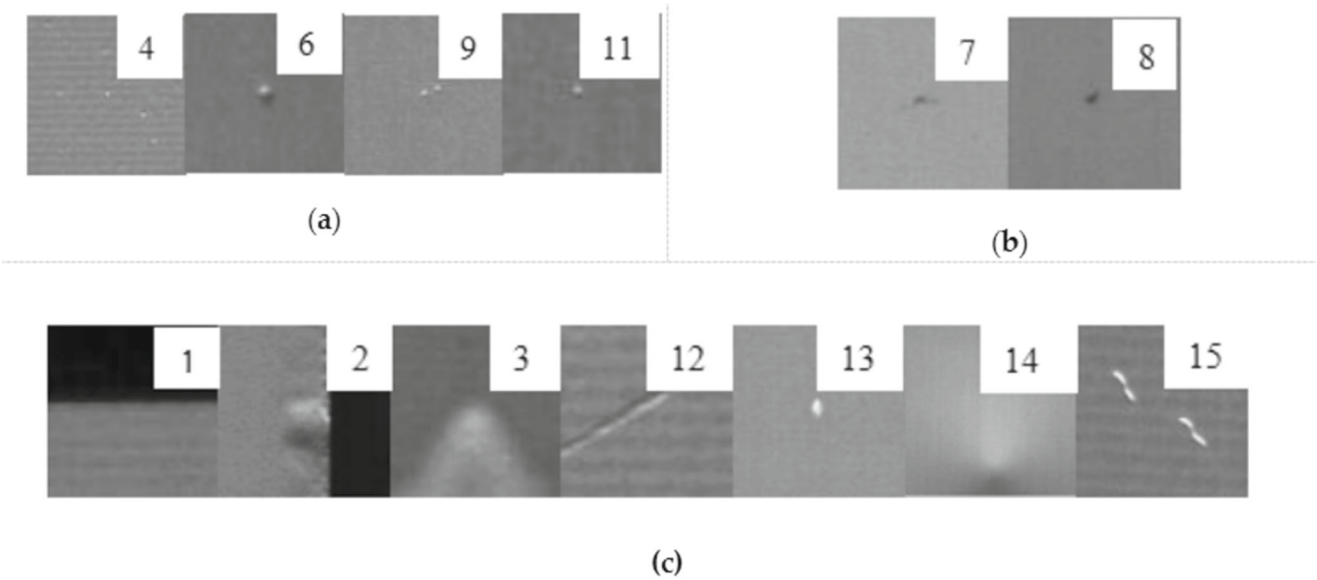



**Fig. 13** Image sample analysis. **a** Defect types No. 4, 6, 9, and 11, which share similar features; **b** Defect types No. 7 and No. 8, which have similar features; **c** Defect types No. 1, 2, 3, 12, 13, 14, and 15, which have distinct features

No. 6, type No. 7, type No. 8, type No. 9 and type No. 11. Through the manual analysis of these image samples, we find that some samples with defect type No. 4, type No. 6, type No. 9 and type No. 11 are similar to each other, resulting in confusion and misclassification. Additionally, some samples with defect type 7 and type 8 share similar features. In contrast, the images with obvious features obtain satisfactory accuracy in every network, such as type No. 1, type No. 2, type No. 3, type No. 12, type No. 13, type No. 14 and type No. 15. These samples are illustrated in Fig. 13. Therefore, we conclude that labeled sample images with distinct features are crucial to achieving high classification accuracy.

We also provide other evaluation metrics for the four networks, such as precision, recall rate and F1-score in Table 5.

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{8}$$

where, TP, FN, TN, and FP represent the numbers of true positives, false negatives, true negatives, and false positives, respectively. These indicators reflect the quality of the model and its generalization capabilities. It can be seen from the table that our model has achieved the best indicators compared with other benchmark networks.

The average computational time for detecting a CCL image is also depicted in Fig. 14. The results are 0.01 seconds for MobileNet-v2, 0.02 seconds for Inception-v3, 0.018 seconds for ResNet-50, and 0.016 seconds for our proposed network. Obviously, MobileNet-v2 achieves the best computation speed results, and our proposed method ranks second.

In addition, we illustrate the five different stages of feature maps generated during the training process in Fig. 14. As shown in Fig. 15, a very important and universal characteristic of representations learned by deep neural networks is that as the depth of the layers increases, the features extracted by the layers become increasingly abstract. Higher-level activations carry less

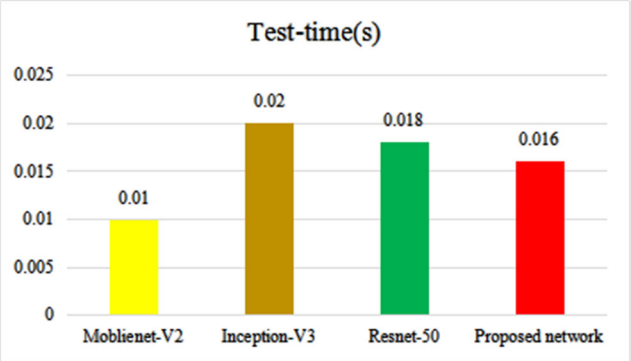



**Fig. 14** The computation cost of four models

information about the input (irrelevant information) and more information about the target (defect information). A deep neural network effectively acts as a pipeline for information extraction. Raw data are input and repeatedly transformed so that irrelevant information is filtered out, whereas useful information is amplified and refined (information of defects).

Compared with the benchmark networks, our proposed network achieves the highest accuracy and suboptimal computation speed. The lightweight network MobileNet has a lower computation cost than our proposed network. However, its accuracy is also lower than that of our proposed network. Our proposed network has a lower computation cost and a higher accuracy than Inception-v3 and Resnet-50. Additionally, our proposed network has achieved a balance between high accuracy and high detection speed. The high detection speed is obtained by using the squeeze-and-expand structure and a 3*3 depthwise separable convolution. The squeeze-and-expand structure decreases 4 times the number of input channels to 3*3 depthwise separable convolutions, resulting in a significant reduction in the calculation parameters. Furthermore, the use of 3*3 depthwise separable convolutions reduces the calculation amount by 8 to 9 times compared with standard convolutions. The high accuracy of our proposed network is achieved by adding more residual blocks into the network while sacrificing less efficiency, as well as the use of an improved squeeze-and-excitation block to improve network performance.

**Table 5** Other evaluation metrics

| | MobileNet-v2 | Inception-v3 | Resnet-50 | Proposed network |
|---|---|---|---|---|
| Precision | 0.98936 | 0.99075 | 0.98562 | 0.99135 |
| Recall rate | 0.98931 | 0.99072 | 0.98527 | 0.99132 |
| F1-score | 0.98931 | 0.99069 | 0.98533 | 0.99132 |

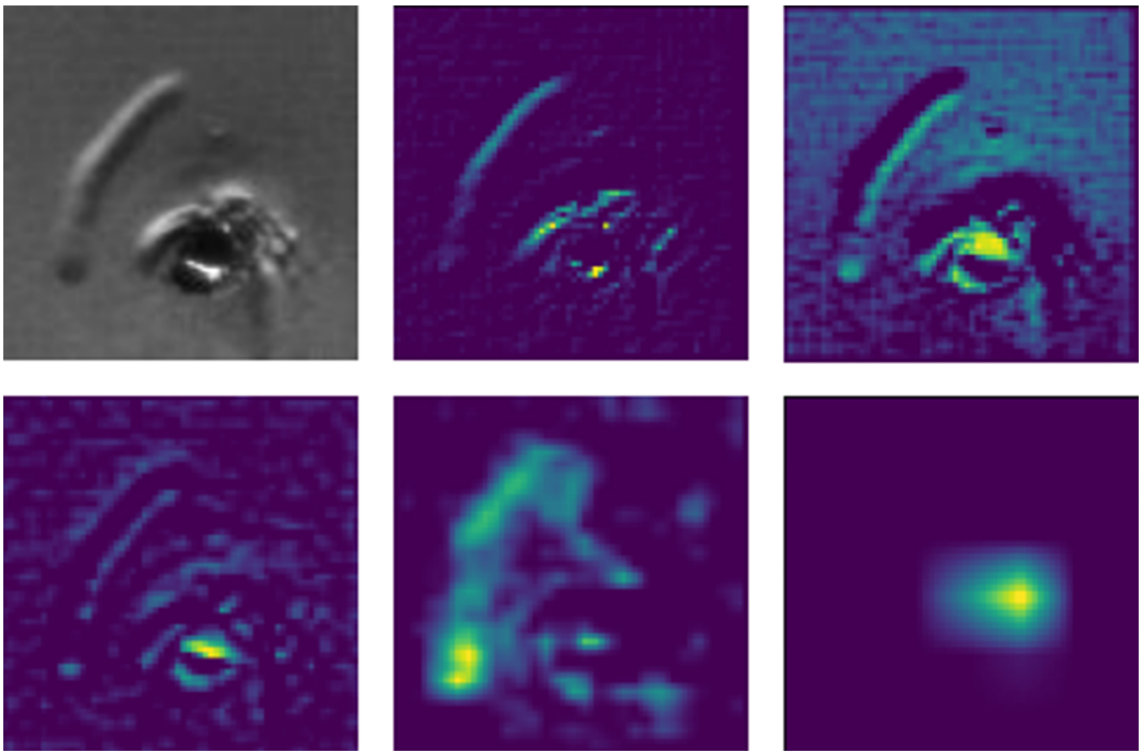


**Fig. 15** Visualization of the feature maps

## 5 Conclusions

This paper mainly studied deep learning technology for the CCL machine vision-based surface defect detection process. We proposed a highly efficient and accurate convolutional neural network to realize accurate and fast CCL defect detection. Our proposed approach makes four contributions. First, we introduce the depthwise separable convolution to reduce the calculation time. Second, we improve the squeeze-and-excitation block to improve network performance. Third, we introduce the squeeze-and-expand mechanism to save computation cost. Fourth, we employ a smoother activation function (Mish) to allow information to flow better. The proposed network is compared with benchmark CNNs, including Inception, ResNet and MobileNet. The experimental results show that our proposed efficient network has achieved a good balance of accuracy and speed and has been chosen for CCL online defect detection since it can meet the requirements of rapid and accurate real-time detection. Our future research will focus on network investigation by applying model compression technology to further improve the calculation efficiency while ensuring accuracy. Additionally, we will broaden our deep learning technology application research in the field of machine vision-based surface inspection.

**Acknowledgements** The authors would like to express their appreciation to the developers of the Keras framework and the developers of classical CNNs, including ResNet, MobileNet and Inception.

## References

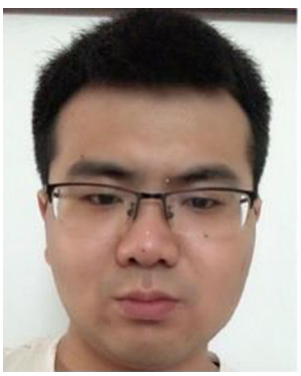
1. Bo T, Jianyi K, Shiqian W, et al. (2017) Review of machine vision surface defect detection. Chinese Journal of Image and Graphics 22:1640–1663
2. Li S, Jing Y, Zheng W (2018) Review of the development and application of defect detection technology. Journal of Automation 15:55–58
3. Huangpeng Q, Zhang H, Zeng X, et al. (2018) Automatic visual defect detection using texture prior and low-rank representation. IEEE Access 6:37965–37976
4. Ojala T, Harwood D (1996) A comparative study of texture measures with classification based on feature distributions. Pattern Recogn 29:51–59
5. Liu K, Wang H, Chen H, et al. (2017) Steel surface defect detection using a new Haar-Weibull-Variance model in unsupervised manner. IEEE Transactions on Instrumentation Measurement, pp 1–12
6. Luo Q, Sun Y, Li P, et al. (2018) Generalized completed local binary patterns for time-efficient steel surface defect classification. IEEE Transactions on Automation Science Engineering, pp 1–13
7. Sun X, Gu J, Tang S, et al. (2018) Research progress of visual inspection technology of steel Products-A review. Appl Sci 8:11

8. Li Y, Zhao W, Pan J, et al. (2017) Deformable patterned fabric defect detection with fisher criterion-based deep learning. IEEE Transactions on Automation Science Engineering 14:1256–1264
9. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20:273–297
10. Altman NS (1992) An introduction to kernel and Nearest-Neighbor nonparametric regression. Am Stat 46:175–185
11. Breiman L, Friedman J, Stone C, et al. (1984) Classification and regression trees. CRC Press
12. Yuan ZC, Zhang Z-T, Su H, et al. (2018) Vision-based defect detection for mobile phone cover glass using deep neural networks. Int J Precision Eng Manufac 19:801–810
13. Jang C, Yun S, Hwang H, et al. (2018) A defect inspection method for machine vision using defect probability image with deep convolutional neural network. In: The 14th Asian conference on computer vision, Perth, Australia, pp 2–6
14. Ren R, Hung T, Tan KC (2018) A generic deep learning-based approach for automated surface inspection. IEEE Transactions on Cybernetics 48:929–940
15. Zheng X, Wang H, Chen J, Zheng S, Kong Y (2020) A generic semi-supervised deep learning-based approach for automated surface inspection. IEEE Access 8:114088–114099
16. Soukup D, Mork H (2014) Convolutional neural networks for steel surface defect detection from photometric stereo images. Advanced in Visual Computing, Berlin, Germany, pp 668–677
17. Je KP, Kwon BK, Park J-H, et al. (2016) Machine learning-based imaging system for surface defect inspection. Int J Precision Eng Manufac Green Technol 3:303–310
18. Wang T, Chen Y, Qiao M, et al. (2018) A fast and robust convolutional neural network-based defect detection model in product quality control. The International Journal of Advanced Manufacturing Technology, Berlin, Germany, pp 3465–3471
19. Caggiano A, Zhang J, Alfieri V, et al. (2019) Machine learning-based image processing for on-line defect recognition in additive manufacturing. CIRP Annals-Manufacturing Technology
20. Michalski P, Ruszczak B, Tomaszewski M (2018) Convolutional neural networks implementations for computer vision. In: The 3rd international scientific conference on brain-computer interfaces, Opole, Poland, pp 13–14
21. Rongsheng L, Ang W, Tengda Z (2018) Review of automatic optical (visual) detection technology and its application in defect detection. Journal of Optics 38:23–58
22. Howard AG, Zhu M, Bo C, et al. (2017) MobileNets: efficient convolutional neural networks for mobile vision applications. arXiv:1704.04861
23. Sandler M, Howard A, Zhu M, et al. (2018) MobileNetV2: inverted residuals and linear bottlenecks. arXiv:1801.04381v3
24. Hu J, Li S, Albanie S, et al. (2018) Squeeze-and-excitation networks. arXiv:1709.01507
25. Forrest N, Iandola SH, et al. (2017) Squeezenet: alexnet-level accuracy with 50x fewer parameters and <0.5mb model. arXiv:1602.07360
26. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
27. Misra D (2019) Mish: a self regularized non-monotonic neural activation function. arXiv:1908.08681
28. Sifre L (2014) Rigid-motion scattering for image classification. Dissertation, Polytechnique
29. Szegedy C, Vanhoucke V, Ioffe S, et al. (2016) Rethinking the inception architecture for computer vision. In: 2016 IEEE conference on computer vision and pattern recognition, Las Vegas, NV, pp 2818–2826
30. Deng W, Dong R, Socher L, et al. (2009) ImageNet: a large-scale hierarchical image database. In: CVPR
31. Tan C, Sun F, Kong T, et al. (2018) A survey on deep transfer learning. In: The 27th international conference on artificial neural networks, Rhodes, Greece, pp 4–7
32. Liu S, Tian G, Xu Y (2019) A novel scene classification model combining ResNet based transfer learning and data augmentation with a filter. Neurocomputing 338:191–206
33. Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. arXiv:1502.03167




**Xiaoqing Zheng** was born in Zhejiang, China in 1981. She received the B.S. and M.S. degrees in control science and engineering from Zhejiang University, China, in 2006. From April 2006 to October 2008, she was a Senior Research Engineer with Honeywell (China) Co., Ltd. Since November 2008, she has been an Assistant Researcher with Automation College of Hangzhou Dianzi University. Her research interests include industrial process modeling and optimization, machine vision based surface inspection, deep learning technology.

**Jie Chen** was born in Changzhou, Jiangsu, China in 1996. He received the B.S. degree in Electrical Engineering and Automation, Jiangsu University of Science and Technology, Jiangsu, China, in 2018. He is currently a master candidate of school of automation in Hangzhou Dianzi University, Zhejiang, China. His research interest includes computer vision and deep learning application.



**Hongcheng Wang** was born in Bozhou, Anhui Province, China in 1993. He received the B.S. degree in Automation major from Anhui University of Engineering, in 2018.

From 2018 to 2019, He is a graduate student at Automation school of Hangzhou Dianzi University. His research direction is semi-supervised learning for industrial application.

**Song Zheng** was born in GuZhou, China in 1982. He received the B.S., M.S. and Ph.D. degrees in control science and engineering from Zhejiang University, China, in 2008. He is currently an Associate researcher with Automation College of Hangzhou Dianzi University. His research interests include industrial automation, modeling and optimization.

**Yaguang Kong** received the B.Sc. and Ph.D. degrees from Zhejiang University in 1997 and 2002 respectively. He is currently an associate Professor with School of Automation, Hangzhou Dianzi University. His main research interests include computer vision, deep learning, robot visual SLAM and process automation.