Received April 27, 2020, accepted June 10, 2020, date of publication June 19, 2020, date of current version June 30, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3003588

A Generic Semi-Supervised Deep Learning-Based Approach for Automated Surface Inspection

XIAOQING ZHENG, HONGCHENG WANG, JIE CHEN, YAGUANG KONG, (Member, IEEE), AND SONG ZHENG
School of Automation, Hangzhou Dianzi University, Hangzhou 310018, China

Corresponding author: Xiaoqing Zheng (zhengxiaoqing@hdu.edu.cn)

This work was supported by the National Natural Science Foundation of China (Grant no. U1609212 and no. 61304211), the Belt and Road International S&T Cooperation Projects of Zhejiang Province (Grant No. 2019C04021), and the Public Technology Research Project of Zhejiang Province (Grant No. LGG20F030002).

ABSTRACT Automated surface inspection (ASI) is critical to quality control in industrial manufacturing processes. Recent advances in deep learning have produced new ASI methods that automatically learn high-level features from training samples while being robust to changes and capable of detecting different types of surfaces and defects. However, they usually rely heavily on manpower to collect and label training samples. In this paper, a generic semi-supervised deep learning-based approach for ASI that requires a small quantity of labeled training data is proposed. While the approach follows the MixMatch rules to conduct sophisticated data augmentation, we introduce a new loss function calculation method and propose a new convolutional neural network based on a residual structure to achieve accurate defect detection. An experiment on two public datasets (DAGM and NEU) and one industrial dataset (CCL) is carried out. For public datasets, the experimental results are compared against several best benchmarks in the literature. For the industrial dataset, the results are compared against deep learning methods based on benchmark neural networks. The proposed method achieves the best performance in all comparisons. In addition, a comparative experiment of model performance given a different number of labeled samples is conducted, demonstrating that the proposed method can achieve good performance with few labeled training samples.

INDEX TERMS Automated surface inspection, defect detection, deep learning, machine vision, MixMatch, semi-supervised learning.

I. INTRODUCTION

To meet the ever-increasing quality standards of industrial manufacturing processes, machine vision systems [1] are used for automated surface inspection (ASI) to automatically check the surface of a finished product for defects such as stains, scratches, holes, pits, and bumps. Compared with manual inspection methods, machine vision systems have the advantages of high efficiency, high accuracy, high speed, and continuous detection and have been widely used in industrial quality control. Machine vision-based ASI mainly includes a process of image acquisition through an optical system [2] and a defect detection process based on acquired images.

Traditional ASI algorithms can be divided into four categories [3]: statistical methods, spectral methods, model-based methods and learning-based methods. Statistical methods utilize first-order or second-order statistics to extract defect features. Popular statistical methods include histogram properties [4], co-occurrence matrix [5] and local binary pattern [6]. The spectral methods transform signals from the spatial domain to the frequency domain for defect identification through Fourier transform [7], wavelet transform [8] and Gabor filters [9]. Model-based methods capture features and identify defects by constructing an image model. The classical model-based methods for defect detection include Markov random field (MRF) [10], autoregressive models [11] and the texems model [12]. Learning-based methods train a system to classify defects by using pattern recognition algorithms such as support vector machines (SVMs) [13], artificial neural networks (ANNs) [14], and k-nearest neighbors (k-NNs) [15]. These four types of methods have a long history of research, each with its own merits. They are usually implemented in a two-stage manner, namely feature extraction and defect classification, and used together in a hybrid mode. For instance,

The associate editor coordinating the review of this manuscript and approving it for publication was Jenny Mahoney.

Çelik *et al.* [16] developed a fabric defect detection method that includes a feature extraction process based on wavelet transform, and then a defect classification process based on co-occurrence matrix and feed-forward neural network.

As the aforementioned ASI methods usually include feature extraction, human experts must design specific rules and tune many parameters to extract desired features. Therefore, human expertise is regarded as the key to the success of ASI [17]. In addition, these methods are sensitive to changes in application conditions and work well only under certain conditions. Recent advances in deep learning [18] have produced novel ASI methods that automatically learn high-level features from training samples and classify defects at the same time without the need to manually design feature sets. Furthermore, they are robust to variations and versatile, allowing inspection of different types of surfaces and defects.

Deep learning-based ASI methods can be classified into three paradigms: supervised learning, unsupervised learning and semi-supervised learning. Supervised learning is the most widely used paradigm, often using deep convolutional neural networks (CNNs) for defect detection. Supervised learning methods based on CNN can achieve high defect detection accuracy given a large number of training data (Kim *et al.* [19], Park *et al.* [20], Li *et al.* [21], Nakazawa and Kulkarni [22], Jeyaraj and Samuel Nadar [23], Liu *et al.* [24], Saiz *et al.* [25], Zhang *et al.* [26], Soukup and Huber-Mork [27]). However, the disadvantage is that they rely heavily on manpower to collect and label training samples. The lack of a large number of labeled samples can be alleviated through unsupervised or semi-supervised learning methods. A literature survey indicates that popular methods of deep unsupervised learning for ASI are deep autoencoders (Mei *et al.* [28], Mujeeb *et al.* [29], Li *et al.* [30]) and generative adversarial networks (GAN) (Zhai *et al.* [31]). The autoencoder is a typical unsupervised learning algorithm based on two neural networks called encoder and decoder. GAN is an unsupervised learning framework that contains a generative model and a discriminative model. GAN can also be extended for semi-supervised learning. The disadvantage of unsupervised learning is that it is generally not as reliable or accurate as supervised learning, while semi-supervised learning provides a solution that combines supervised learning with unsupervised learning in a framework.

Semi-supervised learning can achieve similar or even better precision than supervised learning but uses fewer labeling samples. However, the current research and application of semi-supervised learning-based ASI methods are rare. A common approach is to use GAN for semi-supervised ASI. For instance, Di *et al.* [32] proposed a semi-supervised learning method based on a convolutional autoencoder (CAE) and semi-supervised GAN to classify surface defects of steel. CAE was trained through unlabeled data and used as a feature extractor, while GAN was introduced for semi-supervised learning to further improve the generalization ability.

In our work, a generic semi-supervised deep learning approach that requires a small quantity of labeled data based on MixMatch [33] is proposed for ASI. MixMatch can be regarded as a sophisticated data augmentation method, which follows the consistency regularization rule that the class of unlabeled data should remain unchanged after augmentation. While following the MixMatch rules to conduct sophisticated data augmentation, we introduce a new method of loss function calculation, employ cutout technology for data augmentation, and propose a new convolutional neural network based on a residual network structure to achieve accurate ASI. An experiment on two public datasets (DAGM and NEU) and one industrial dataset (CCL) is carried out. The proposed method achieves the best performance in comparisons with several of the best benchmarks in the literature or with the benchmark deep learning methods. In addition, a comparative analysis of model performance given a different number of labeled samples is conducted, demonstrating that the proposed method can achieve good performance with few labeled training samples.

II. RELATED WORK

This section introduces the related algorithms on which our proposed method is based, including MixMatch, Cutout and ResNet.

A. MIXMATCH

MixMatch [33] is a sophisticated data augmentation method that follows the consistency regularization rule that the class of unlabeled data should remain unchanged after augmentation. MixMatch produces new image samples by mixing labeled samples and unlabeled samples through the MixUp [34] method after data augmentation, label guessing, averaging and sharpening. Given labeled input data X and unlabeled input data U, MixMatch generates labeled data X' and unlabeled data U' with predicted labels. The algorithm is illustrated in Table 1.

TABLE 1. Main algorithm of the MixMatch method.

	MixMatch
1	$\widehat{X}$ = Augment(X): Apply data augmentation to X
2	$\hat{U}$ = Augment(U) for k times: Apply data augmentation to unlabeled data k times
3	$\bar{Q}$ = Average(P_{model}($\hat{U}$)): Predict the labels of $\hat{U}$ and average the k predictions
4	Q = Sharpen ($\bar{Q}$, T): Apply temperature sharpening to $\bar{Q}$
5	W = Shuffle (Concat($\widehat{X}$, $\hat{U}$)): Combine and shuffle labeled and unlabeled data
6	X′ = MixUp($\widehat{X}$, W_x): Apply MixUp to $\widehat{X}$ and the same size W_x selected from W
7	U′ = MixUp($\hat{U}$, W_u): Apply MixUp to U′ and the same size W_u (the rest of W)

where P_{model} is a network calculating the class distributions of input data. T is the sharpening temperature applied in a sharpening function to reduce the entropy of the label distribution. The sharpening function is as follows.

$$Sharpen(\bar{Q}, T)_i = \bar{Q}_i^{1/T} / \sum_j \bar{Q}_j^{1/T} \tag{1}$$

The MixUp function with input items (x_1, p_1) and (x_2, p_2) is calculated through the following equations (2) to (5), outputting (x', p'), where x_1 and x_2 are input data with corresponding labels p_1 and p_2, while x' and p' are the output data and corresponding labels. α is a hyperparameter used for generating the Beta distribution.

$$\lambda = \mathrm{Beta}(\alpha, \alpha) \tag{2}$$
$$\lambda' = \max(\lambda, 1-\lambda) \tag{3}$$
$$x' = \lambda' x_1 + (1-\lambda') x_2 \tag{4}$$
$$p' = \lambda' p_1 + (1-\lambda') p_2 \tag{5}$$

B. CUTOUT

Data augmentation is an effective technique for increasing both the quantity and diversity of data by randomly augmenting it [35]. The basic and common augmentation methods include random cropping, horizontal flipping, and color shifting. The data augmentation method employed in our work is an augmentation method named Cutout [36]. Cutout is a simple regularization technique, which is easily implemented by randomly masking out square regions of an input image. Cutout removes contiguous sections of input images, generating augmented versions of image samples with masked cutout regions. It performs well on vision benchmark datasets and can be used in conjunction with other augmentation methods such as MixMatch.

C. RESIDUAL NEURAL NETWORK

The CNN is the most widely used deep learning algorithm for defect detection due to its stable ability to learn high-level features from training data. CNN has three kinds of layers: convolutional layers, pooling layers and fully connected layers. The convolutional layer extracts features of an input image by applying multiple convolutional kernels. The pooling layer is used for dimensionality reduction and feature selection. The fully connected layer combines the features in a vector and outputs them to the final classifier or other fully connected layers. The fully connected layer can also be replaced by an average pooling layer. Theoretically, features can be enriched by stacking more network layers together, thus achieving better performance. However, as the number of network layers increases, the training accuracy saturates and then degrades rapidly, which is called the degradation problem [37]. This problem can be alleviated by the residual network structure proposed in ResNet [37]. The basic residual building block introduced by ResNet is illustrated in Figure 1. As shown in Figure 1, an identity shortcut connection that skips two weight layers is added to perform identity mapping, and its output is added to the outputs of the stacked two weight layers. Specifically, with an input x, the mapping is recast into F(x)+x, where F(x) denotes the mapping of the two stacked layers. The identity shortcut connections introduce neither extra parameters nor computational complexity [37]. A residual network can reuse the features from previous layers and alleviate the degradation problem by shortcut connections. Therefore, the performance of a residual network can be improved by simply stacking more residual building blocks. Numerous practical applications have proven that deep residual networks are easier to optimize than corresponding networks without shortcut connections.

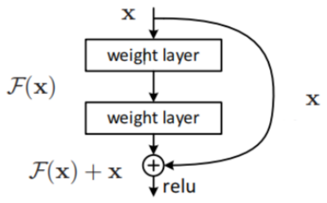

x
weight layer
$\mathcal{F}(\mathbf{x})$
weight layer
x
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

FIGURE 1. Residual building block [37]: two weight layers with an identity shortcut connection.

III. THE PROPOSED METHOD

A generic semi-supervised learning approach for automated surface inspection has been proposed based on MixMatch. The calculation process is demonstrated in Figure 2. Unlabeled image samples (u) are augmented by using cutout twice. Then, the class distributions of unlabeled samples are predicted by convolutional neural network f_θ, followed by a process of averaging, sharpening and concatenating. Additionally, labeled image samples (x, y) are augmented by cutout with labels unchanged. The augmented labeled data and augmented unlabeled data are combined and shuffled and mixed to obtain new data (x', y') and (u', p'), respectively. Then, the new data x' and u' are fed into the convolution neural network f_θ to calculate the class distributions. The calculated distributions f_θ(x') and f_θ(u') are compared with y' and p', respectively, to obtain the supervised loss and unsupervised loss. Where x, x', u' are image data, while y, y' and p' are corresponding class distributions.

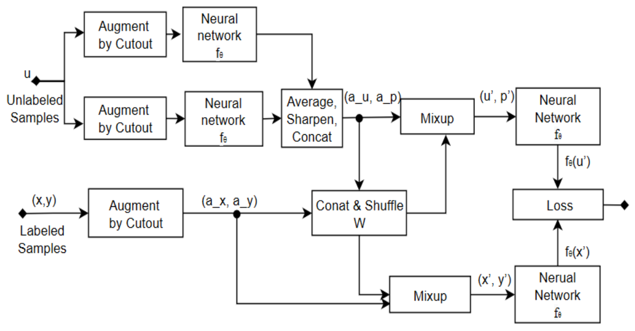


FIGURE 2. Calculation process of the proposed method.

In addition, we introduce a new method of loss function calculation, employ cutout technology for data augmentation, and propose a new convolutional neural network to achieve accurate ASI.

The loss function we propose consists of two items, supervised loss item L_S and unsupervised loss item L_U. The supervised loss item uses the common cross entropy loss function, which calculates the difference in distributions between y' and f_θ(x'). For the unsupervised item, we use a combination of the mean square error L_{MSE} and Kullback-Leibler (KL) divergence L_{KL} to improve the network performance, both of which calculate the difference in distribution between p' and f_θ(u'). A hyperparameter α is used to adjust the proportions of L_{MSE} and L_{KL} in the unsupervised loss term through the Beta distribution. Beta distribution refers to a set of continuous probability distributions defined in the interval (0, 1). The calculation process of the loss function is shown from equations (6) to (13).

$$L_S = H(y', f_\theta(x')) = -\mathbb{E}_{x'\sim y'}[\log(f_\theta(x'))] \tag{6}$$

$$L_{MSE} = \mathbb{E}[(p' - f_\theta(u'))^2] \tag{7}$$

$$L_{KL} = D_{KL}(p'||f_\theta(u')) = \mathbb{E}_{u'\sim p'}[\log(p') - \log(f_\theta(u'))] \tag{8}$$

$$\lambda_1 = \mathrm{Beta}(\alpha, \alpha) \tag{9}$$

$$\lambda_1 = \max(\lambda_1, 1 - \lambda_1) \tag{10}$$

$$L_U = \lambda_1 L_{MSE} + (1 - \lambda_1)L_{KL} \tag{11}$$

$$\lambda_2 = \mathrm{Beta}(\alpha, \alpha) \tag{12}$$

$$\mathrm{loss} = L_S + \lambda_2 L_U \tag{13}$$

The algorithm of the semi-supervised learning approach for ASI is described in Table 2.

TABLE 2. Algorithm of the semi-supervised learning approach for ASI.

```
Require: $(x_i, y_i)$ = image samples $x_i$ with labels $y_i$ in the training set
Require: $u_j$ = unlabeled image samples $u_j$ in the training set
Require: $f_\theta$ = convolutional neural network with trainable parameters $\theta$
Require: Cutout = stochastic data augmentation function
for t in [1, number_of_epochs] do
  for each minibatch B do
    for i = 1 to B do
      $(a\_x_i, a\_y_i)$ = cutout($x_i$): stochastic augmentation of labeled data
      $a\_u1_i$ = cutout($u_i$): stochastic augmentation of unlabeled data
      $a\_u2_i$ = cutout($u_i$): stochastic augmentation of unlabeled data
      $avg\_p_i = (f_\theta(a\_u1_i) + f_\theta(a\_u2_i))/2$: average label predictions
      $u\_p_i$ = Sharpen($avg\_p_i$, T): sharpen average predictions by using (1)
    end for
    (a_u, a_p) = concat((a_u1, u_p), (a_u2, u_p)): combine unlabeled data
    W = shuffle(concat((a_x, a_y), (a_u, a_p)): combine and shuffle data
    $(x', y')$ = Mixup(($a\_x_i$, $a\_y_i$), $W_i$), $i \in (1,...,B)$: use equations (2) to (5)
    $(u', p')$ = Mixup(($a\_u_j$, $a\_p_j$), $W_{B+j}$) , $j \in (1,...,2*B)$: use (2) to (5)
    Ls = CrossEntropy($y'_i$, $f_\theta(x'_i)$), $i \in (1,...,B)$: calculate by using (6)
    $L_{MSE}$ = MSE($p'_j$, $f_\theta(u'_j)$), $j \in (1,...,2\times B)$: calculate by using (7)
    $L_{KL} = D_{KL}(p'_j || f_\theta(u'_j))$, $j \in (1,...,2\times B)$: calculate by using (8)
    Calculate loss using equations (9) to (13)
    Update $\theta$ using optimizer ADAM
  end for
end for
return $\theta$
```

A. CUTOUT METHOD

Cutout is extremely easy to implement and can be used in conjunction with other augmentation methods. Furthermore, it can improve the robustness and overall performance of convolutional neural networks.

Cutout is implemented by randomly masking out one square patch from an image. For large images (such as images in the DAGM and NEU datasets), a square patch with a size of 16 × 16 is employed as the cutout region. For small size images such as the CCL industrial dataset, a square patch with size 4 × 4 is used as the cutout region. When applying a square patch mask, we randomly select a pixel coordinate in the image as the center point and then place the square patch around the center point.

Several augmented samples through cutout are illustrated in Figure 3.

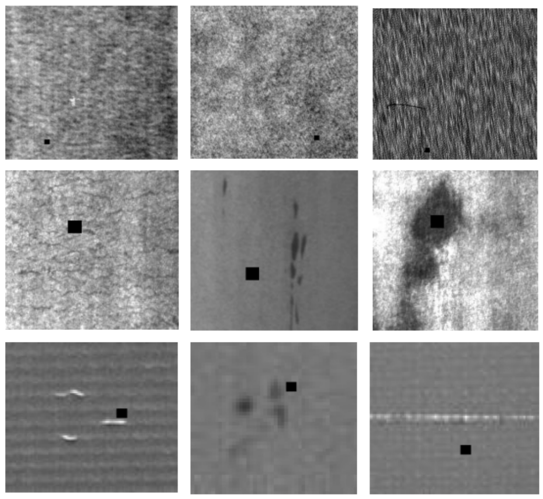


FIGURE 3. Augmented image samples after cutout. The black square patches in the images are the cutout regions. The three images in the first row are from the DAGM dataset, the images in the second row are from the NEU dataset, and the images in the last row are from the CCL dataset.

B. PROPOSED CNN

A deep convolutional neural network based on a residual network structure [37] is proposed. First, a residual building block having two 3 × 3 convolution layers with a parameter-free identity shortcut connection, as shown in Figure 1, is applied for network design. Then, a convolutional neural network f_θ with the required depth and width is introduced to meet the performance demand of surface defect detection by stacking the two-layer residual building blocks.

Specifically, we propose a 43-layer convolutional neural network by stacking two-layer residual building blocks together. Since this is a deep network that is not easy to converge, the residual building blocks are applied as its basis blocks to help the convergence and improve the network performance. In addition, we increase the width of the network to achieve a balance between network depth and width, and to gain higher accuracy as well.

The overall structure of the proposed network is shown in Table 3. To reduce overfitting of the network, dropout and batch normalization are used before each convolution operation. The network structure and calculation process are described as follows.

TABLE 3. Proposed convolutional neural network.

Group name	Output size	Filter shape (kernel size, number of kernels, stride)
Conv1	256 × 256 × 16	3 × 3, 16, stride = 2
Block1	128 × 128 × 32	$\begin{bmatrix} 3\times 3, 32, \text{stride}=2 \\ 3\times 3, 32, \text{stride}=1 \end{bmatrix}$
Block2	128 × 128 × 32	$\begin{bmatrix} 3\times 3, 32, \text{stride}=1 \\ 3\times 3, 32, \text{stride}=1 \end{bmatrix}\times 3$
Block3	64 × 64 × 64	$\begin{bmatrix} 3\times 3, 64, \text{stride}=2 \\ 3\times 3, 64, \text{stride}=1 \end{bmatrix}$
Block4	64 × 64 × 64	$\begin{bmatrix} 3\times 3, 64, \text{stride}=1 \\ 3\times 3, 64, \text{stride}=1 \end{bmatrix}\times 3$
Block5	32 × 32 × 128	$\begin{bmatrix} 3\times 3, 128, \text{stride}=2 \\ 3\times 3, 128, \text{stride}=1 \end{bmatrix}$
Block6	32 × 32 × 128	$\begin{bmatrix} 3\times 3, 128, \text{stride}=1 \\ 3\times 3, 128, \text{stride}=1 \end{bmatrix}\times 3$
Block7	16 × 16 × 256	$\begin{bmatrix} 3\times 3, 256, \text{stride}=2 \\ 3\times 3, 256, \text{stride}=1 \end{bmatrix}$
Block8	16 × 16 × 256	$\begin{bmatrix} 3\times 3, 256, \text{stride}=1 \\ 3\times 3, 256, \text{stride}=1 \end{bmatrix}\times 3$
Block9	8 × 8 × 512	$\begin{bmatrix} 3\times 3, 512, \text{stride}=2 \\ 3\times 3, 512, \text{stride}=1 \end{bmatrix}$
Block10	8 × 8 × 512	$\begin{bmatrix} 3\times 3, 512, \text{stride}=1 \\ 3\times 3, 512, \text{stride}=1 \end{bmatrix}\times 3$
Avg-pool	1 × 1 × 512	8 × 8 average pool

·The size of the input picture is 512 × 512.

·Conv1: a 3 × 3 convolution with 16 filters and stride 2.

·Block1: a residual building block with two 3 × 3 convolution layers.

·Block2: three residual building blocks, each containing two 3 × 3 convolution layers.

·Block3: a residual building block with two 3 × 3 convolution layers.

·Block4: three residual building blocks, each containing two 3 × 3 convolution layers.

·Block5: a residual building block with two 3 × 3 convolution layers.

·Block6: three residual building blocks, each containing two 3 × 3 convolution layers.

·Block7: a residual building block with two 3 × 3 convolution layers.

·Block8: three residual building blocks, each containing two 3 × 3 convolution layers.

·Block9: a residual building block with two 3 × 3 convolution layers.

·Block10: three residual building blocks, each containing two 3 × 3 convolution layers.

·An average pool with kernel size 8 × 8 and stride 1.

·A fully connected layer and classifier to accomplish classification.

The proposed neural network applies a small convolution kernel number of 16 in the first layer. In the following eight-layer convolutional network from Block1 to Block2, the number of kernels has only doubled to 32. Then, the number of kernels doubles every eight layers. This strategy of slowly increasing the number of convolution kernels can reduce the network parameters and the amount of calculation. The proposed network has reached a balance of appropriate width and depth to maximize the network performance and efficiency. In addition, the residual network structure is adopted as the fundamental architecture. Therefore, even if the depth reaches 43 layers, it can still be easily optimized.

For the DAGM dataset, whose image size is 512 × 512, we use exactly this network. However, for the NEU dataset and CCL dataset, slight modifications are made to make the network adaptive to the size of the input image. For the NEU dataset with an image size of 200 × 200, Block9 and Block10 are removed from the network. For the CCL dataset with an image size of 32 × 32, Block7, Block8, Block9 and Block10 are removed from the network, and the strides of Conv1 and Block1 are set to 1 as well.

IV. DATA DESCRIPTION

A. DAGM TEXTURE DATASET

The DAGM texture database [38] is a synthetic benchmark dataset for defect detection on statistically textured surfaces. The image samples in DAGM are artificially generated, but they are similar to real world problems. It consists of six types of artificially generated texture images. Each type has 1,000 nondefective images and 150 defective images and has been randomly split into a training and testing subset of almost equal size. Specifically, the training set contains 3,450 images, including 3,004 defect-free image samples and 446 defective image samples.

TABLE 4. DAGM texture dataset.

	Training set		Testing set	
Subclass	Defect-free	Defective	Defect-free	Defective
Class 1	496	79	504	71
Class 2	509	66	491	84
Class 3	509	66	491	84
Class 4	493	82	507	68
Class 5	505	70	495	80
Class 6	492	83	508	67

The DAGM dataset is challenging for two reasons [39]: the defective background texture in the same class varies greatly, and some of the defect regions are very small or similar to the background texture.

B. NEU STEEL DATASET

The NEU surface defect dataset [40] contains six kinds of typical surface defects of hot-rolled steel strips, including rolled-in scale (RS), patches (Pa), crazing (Cr), pitted surface (PS), inclusion (In) and scratches (Sc). The database contains 1,800 grayscale images, with 300 samples in each class. We randomly choose 70% of the samples from each class as the training set, and the rest are for the testing set.

C. INDUSTRIAL CCL DATASET

An industrial dataset with 12,380 image samples was collected from an industrial copper clad laminate (CCL) machine vision inspection system. CCL is widely used in the electronics industry to manufacture printed circuit boards (PCBs). The surface defects of CCL include scratches, oil stains, pinholes, and inclusions. They are divided into two categories in industrial applications, namely, severe defects and non-severe defects. Severe defects are intolerable for quality control, while non-severe defects are acceptable. Therefore, the collected images were manually labeled as two categories. Seventy percent of the images were randomly chosen as the training set, while the remaining 30% were used as the testing set. The dataset information is shown in Table 5.

TABLE 5. Industrial ccl dataset.

Subclass	Training set	Testing set
Non-severe defects	3,598	1,542
severe defects	5,068	2,172

Several image samples of two categories of the CCL dataset are illustrated in Figure 4 and Figure 5.

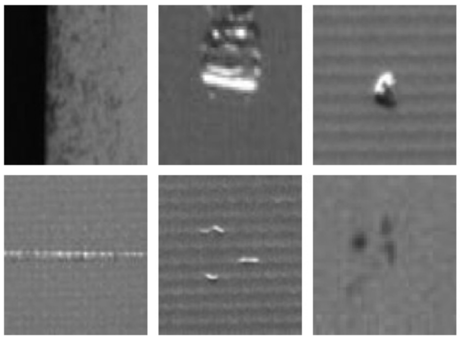


FIGURE 4. Several severe defective image samples.

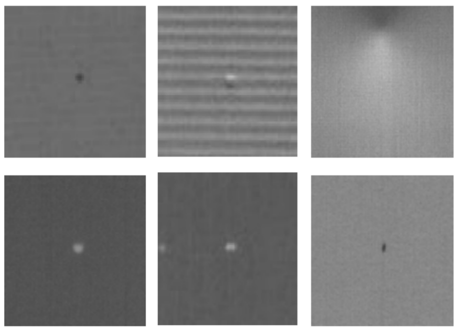


FIGURE 5. Several non-severe defective image samples.

V. EXPERIMENTS AND COMPARISONS

The experiments are carried out on an NVIDIA DXG station configured with an NVIDIA Tesla V100 GPU and 32 GB memory. The configuration and training are realized by using the PyTorch deep learning library in the Python language. The performance of the methods is evaluated on the testing set of the DAGM, NEU and CCL datasets.

The training parameters are set as follows.

·The maximum epoch is set to 1024.

·The batch size is set to 32, which means that 32 samples are taken for training each time. As we know, large batch tend to converge to sharp minimizers of the training and testing functions, resulting in poorer generalization [41]. Our comparative experiments have verified that a batch size of 32 can achieve good ASI accuracy results. For the CCL dataset, the accuracy of the CCL testing set is 93.43%, 95.67%, 95.45% and 94.89% when the batch size is set to 16, 32, 64, and 100, respectively. At the same time, the batch size of 32 does not require high GUP memory.

·The Adam optimizer is applied and a learning rate of 0.002 is adopted.

·The Xavier initialization method [42] is applied to initialize the weights of our proposed neural network. The initial weights are randomly selected from a normal distribution with a mean of 0 and a standard deviation of sqrt (2/(n_in+n_out)), where n_in and n_out are the number of input and output neuros of each layer, respectively. This initialization method aims to keep activation variance and back-propagated gradient variance of each layer unchanged.

In addition, the hyperparameter α used for the Beta distribution in equation (9) and equation (12) is set to 0.75. The hyperparameter α acts as a regulator to adjust the proportions of the two items L_{MSE} and L_{KL} in L_U in equation (11), and the proportions of the two items L_S and L_U in loss in equation (13). We hope that there is a certain difference between the proportions of the two items, rather than equal proportions. According to the probability density function of Beta distribution shown in Figure 6, the value of α should be less than 1.0 to minimize the possibility of having a Beta (α, α) distribution of 0.5. At the other hand, we also don't expect a huge difference in the proportions of the two items, otherwise it may cause the entire loss function unbalanced. From this perspective, $\alpha = 0.75$ is much better than $\alpha = 0.25$ as shown in Figure 6. And according the variance of Beta distribution shown in equation (14), when α increases, the variance of $X \sim \text{Beta}(\alpha, \alpha)$ decreases. A larger α is preferable for reducing the variance. Furthermore, comparative experiments related to α on the CCL dataset is conducted. The accuracy of the CCL testing set is 94.72%, 94.96%, 95.67% and 94.89%

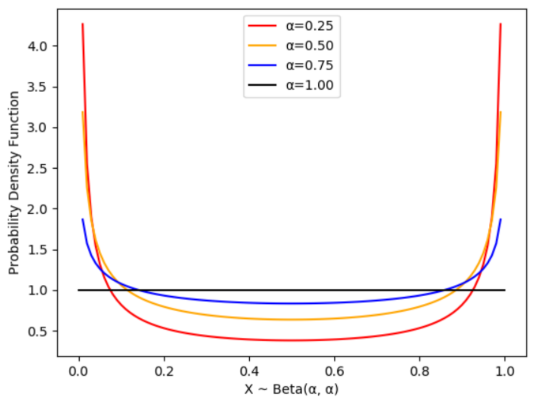


FIGURE 6. Probability density function (PDF) of $X \sim$ Beta (α, α).

when α is set to 0.25, 0.50, 0.75 and 1.00, respectively. Therefore, we believe that 0.75 is an appropriate value for α.

$$\text{Variance}(X) = \alpha\beta / \{(\alpha+\beta)^2(\alpha+\beta+1)\},\ X \sim \text{Beta}(\alpha, \beta) \tag{14}$$

A. DAGM TEXTURE DATASET

A comparative analysis of model performance given different percentages of labeled training samples is carried out. 10% to 70% of labeled samples are randomly selected from the DAGM training set, while the labels of the remaining samples are removed, obtaining unlabeled training data. The true positive rate (TPR), true negative rate (TNR) and average accuracy are employed as evaluation metrics. TPR is the percentage of defect-free samples correctly classified as defect-free. TNR is the percentage of defective samples correctly classified as defective. The performance on the DAGM testing set is shown in Table 6.

When given 10% of labeled training samples, which actually only contain 345 labeled samples, the overall accuracy is 90.75%. When 30% of labeled training samples are given, which contains 1,035 labeled samples, the overall accuracy is 93.42%. The TNRs for Classes 2, 3, 4 and 5 are zero because there are too few labeled defect samples in each class. For instance, when 10%-labeled samples are given, only 8 labeled defective samples in Class 4 participate in training. When the percentage of labeled training samples increases to 50%, an overall accuracy of 98.81% is obtained. When 60% or 70% of labeled training samples are given, very good performance is achieved. Given 70% of labeled samples (the number of labeled training samples is 2,415), which achieves a high accuracy of 99.83%, and we compare the result of this scenario against the best benchmarks in the literature. According to our best knowledge, this is better than the best benchmark in the literature. The results in Table 6 demonstrate that our proposed method can achieve very high accuracy with a small amount of labeled training data, thereby verifying the effectiveness of the algorithm. However, if too few defect samples are labeled, accuracy can suffer. There should be a

TABLE 6. Results when given different percentages of labeled samples of training set.

Class	10% Labeled	30% Labeled	50% Labeled	60% Labeled	70% Labeled
TPR (%)					
Class 1	100	100	100	100	100
Class 2	100	100	100	100	100
Class 3	100	100	100	100	100
Class 4	100	100	100	100	99.21
Class 5	100	100	100	100	100
Class 6	100	100	100	100	100
TNR (%)					
Class 1	100	100	100	100	100
Class 2	0	100	100	100	100
Class 3	0	0	98.81	94.05	100
Class 4	0	8.82	44.12	98.53	98.53
Class 5	0	0	98.75	100	100
Class 6	95.52	98.51	98.51	98.51	98.51
Average Accuracy (%)					
	90.75	**93.42**	**98.81**	**99.80**	**99.83**

balance between the small size of labeled training data and high accuracy.

The best benchmark in the literature for the DAGM dataset is a twofold joint detection CNN network [39]. It is a CNN-based supervised learning method that uses 4,83 training samples and achieves an accuracy result of 99.8%. Our proposed method uses half of the training samples as in CNN [39] but achieves better accuracy. CNN [43] is also a deep learning-based method that utilizes 70% of 1,299,200 samples obtained after data augmentation for training. Other methods such as SIF [44] and Weibull [45] listed in Table 7 are traditional defect inspection methods other than deep learning methods. Compared with the supervised learning methods CNN [39] and CNN [43], our proposed method achieves better accuracy results with much fewer labeled training samples. Compared with traditional methods such as SIF [44] and Weibull [45], our proposed method also achieves better accuracy results.

The feature visualization of the proposed network on DAGM samples is demonstrated in Figure 7. As illustrated in Figure 7, the convolutional neural network attempts to detect high-level features that are useful for final defect classification as its layers become deeper. It is hard for humans to determine what is illustrated in the last column of this figure, but it is easy to determine from the third column that the network attempts to detect the defective patterns of the three image samples.

B. NEU STEEL DATASET

A comparative analysis of model performance given different percentages of labeled training samples is also carried

TABLE 7. Results of our proposed method and several benchmarks in the literature.

Class	Our Method	CNN [39]	CNN [43]	SIF [44]	Weibull [45]
TPR (%)					
Class 1	100	100	100	98.9	87.0
Class 2	100	100	100	95.7	-
Class 3	100	100	95.5	98.5	99.8
Class 4	99.21	100	100	-	-
Class 5	100	99.7	98.8	98.2	97.2
Class 6	100	100	100	99.8	94.9
TNR (%)					
Class 1	100	100	100	100	98.0
Class 2	100	100	97.3	91.3	-
Class 3	100	100	100	100	100
Class 4	98.53	93.2	98.7	-	-
Class 5	100	100	100	100	100
Class 6	98.51	100	99.5	100	100
	99.83	**99.8**	**99.2**	**98.2**	**97.1**

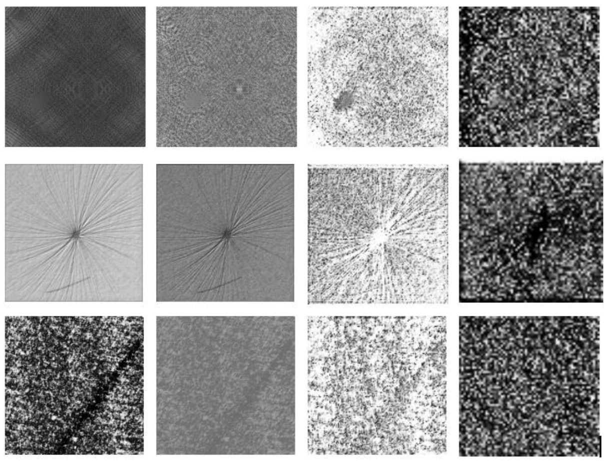


FIGURE 7. Visualization of features in different layers of three defective samples: the first column is the original image; the second column shows the feature maps from the first convolutional layer (Conv1); the third column illustrates the feature maps from the ninth convolutional layer (Block2); and the last column demonstrates the feature maps from the 17th convolutional layer (Block4).

out. Ten, 20, and 30% of the labeled samples are randomly selected from the NEU training set, while the labels of the remaining samples are removed to obtain unlabeled training data. Specifically, the training set contains 1260 image samples (210 samples for each class). The experimental results on the NEU testing set are shown in Table 8. Accuracy is used for the evaluation metrics. For each class, accuracy equals the ratio of the number of image samples classified correctly to the total number of image samples in this class.

When given 10% of labeled training samples, which actually contain 126 the labeled samples (with 21 labeled samples in each class), the overall accuracy is 99.63%.

TABLE 8. Accuracy (%) results when given different percentages of labeled training samples.

Class	10% Labeled	20% Labeled	30% Labeled
RS	100	100	100
Pa	97.78	100	100
Cr	100	100	100
PS	100	100	100
In	100	100	100
Sc	100	98.89	100
Overall	**99.63**	**99.81**	**100**

When 30% of the labeled training samples are given, which contain 378 labeled samples (with 63 labeled samples in each class), the overall accuracy is 100%. The results show that our proposed method performs very well in this dataset and can achieve very high accuracy with an extremely small quantity of labeled training data.

The classification results of our proposed method using 378 labeled training samples (the accuracy is 100%) are compared with a traditional detection method based on local binary patterns and three deep-learning based methods. Their performances on the NEU dataset are shown in Table 9. The accuracy results of these benchmark methods listed in Table 9 are calculated from the confusion matrix published in corresponding papers. A generalized completed local binary patterns (GCLBP) method [46] achieves an overall accuracy of 99.11% on the NEU dataset, which uses 900 images for training. An end-to-end surface defect inspection method based on deep CNN [47] achieves an overall accuracy of 99.0%. Its NEU training dataset is enlarged by a factor of five through data augmentation. Another deep CNN-based supervised learning method [25] achieves a high classification accuracy of 99.95% based on an expanded NEU dataset. It also compared its results with the existing 17 related traditional defect detection methods and outperformed all these methods. Compared with the best benchmark [25], our proposed method achieves better accuracy with much fewer labeled samples. In addition, a recent

TABLE 9. Accuracy (%) of our proposed method and several benchmarks in the literature.

Class	Our Method	GCLBP [46]	CNN [47]	CNN [25]	PLCNN [48]
RS	100	100	100	100	-
Pa	100	98.67	98.67	100	-
Cr	100	100	99.33	100	-
PS	100	98.00	98.00	100	-
In	100	98.67	98.67	100	-
Sc	100	99.33	99.33	99.67	-
Overall	**100**	**99.11**	**99.0**	**99.95**	**90.7**

semi-supervised deep learning-based method PLCNN [48] is investigated. PLCNN achieves an accuracy of 90.7%. Its NEU training set contains 1500 image samples. Each type of the NEU defects retains 50 labeled samples and removes the remaining labels. Although the training set of our proposed method contains only 1260 images, when 21 labeled samples are retained for each type of the NEU defects and the remaining labels are deleted, a higher accuracy of 99.63% can still be achieved. Therefore, our proposed method achieves the best accuracy with the least number of labeled samples used in the NEU dataset.

The feature visualization of our proposed network on NEU samples is illustrated in Figure 8. As illustrated in Figure 8, the convolution neural network attempts to detect high-level features that are useful for final defect classification as its layers become deeper. The high-level features here are the defective patterns of the image samples. Proper detection of defective patterns is essential for achieving accurate classification results.

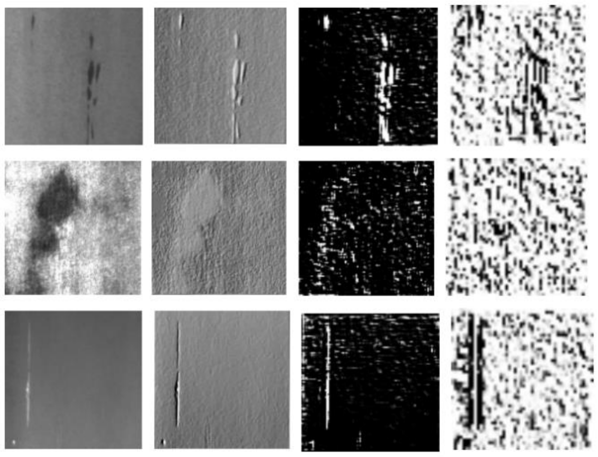


FIGURE 8. **Visualization of features in different layers of three defective samples (In, Pa, Sc): the first column is the original image; the second column shows the feature maps from the first convolutional layer (Conv1); the third column illustrates the feature maps from the ninth convolutional layer (Block2); the last column demonstrates the feature maps from the 17th convolutional layer (Block4).**

C. INDUSTRIAL CCL DATASET

A comparative analysis of model performance when given different percentages of labeled training samples is also performed for the CCL dataset. Ten, 30 and 50% of the samples in the CCL training set are randomly selected for labeled training samples. The labels of the remaining samples in the CCL training set are removed, obtaining unlabeled training data. Accuracy is also used for the evaluation metrics. The performance on the CCL testing set is demonstrated in Table 10.

When the percentage of labeled samples increases from 10% to 50%, the overall accuracy increases from 91.46% to 95.67%. However, using more labeled samples does not help to improve accuracy. Therefore, we conducted a comparison

TABLE 10. **Accuracy (%) when given different percentages of labeled training samples.**

Class	10% Labeled	30% Labeled	50% Labeled
Severe defects	91.89	95.98	96.24
Non-severe defects	91.16	93.69	95.26
Overall	**91.46**	**94.64**	**95.67**

experiment with a popular semi-supervised method called Bad GAN [49] and two supervised deep learning methods based on classical convolutional neural networks ResNet-50 and ResNet-101. We realized supervised learning based on ResNet-50 and ResNet-101 for CCL defect classification using the whole CCL training set. The accuracy results of ResNet-50 and ResNet-101 are 94.10% and 94.64%, respectively, lower than that of our method 95.67%. And we realized semi-supervised learning based on Bad GAN, also using 10%, 30% and 50% of the labeled samples of the CCL training set. Its overall accuracy is 88.58%, 89.96% and 90.98%, respectively. Bad GAN's accuracy is lower than our proposed method even using more labeled samples. The accuracy results of the four methods are demonstrated in Figure 9.

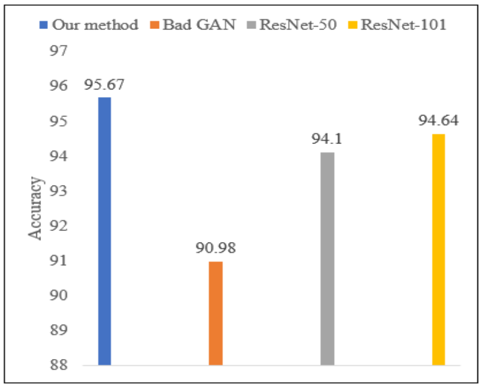
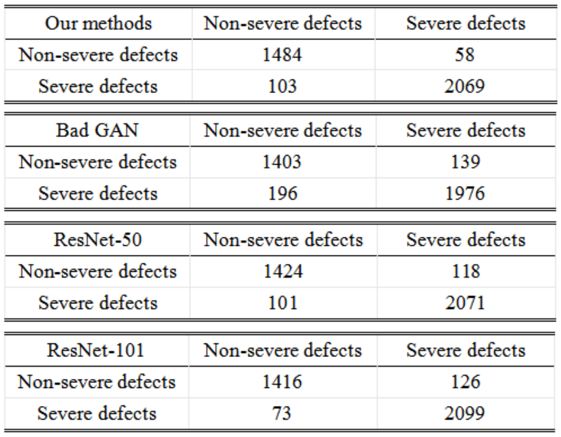


FIGURE 9. **Accuracy results of our proposed method (using 50% of the labeled training samples), Bad GAN (using 50% of the labeled training samples), ResNet-50 and ResNet-101 (using 100% of the labeled training samples).**

In addition, the confusion matrix of the four methods is demonstrated in Figure 10. In the confusion matrix, the first column is the true classes, and each row corresponds to the predicted classes. The diagonal cells correspond to items that are correctly classified.

The classification accuracy is not particularly high for either our proposed method or the benchmark deep learning methods Bad GAN, ResNet-50 and ResNet-101. The possible reason is that the difference in characteristics between severe defects and non-severe defects is not always obvious, so sometimes they are easily confused with each other.

The feature visualization of our proposed network on CCL samples is illustrated in Figure 11. As illustrated in Figure 11,



Our methods	Non-severe defects	Severe defects
Non-severe defects	1484	58
Severe defects	103	2069

Bad GAN	Non-severe defects	Severe defects
Non-severe defects	1403	139
Severe defects	196	1976

ResNet-50	Non-severe defects	Severe defects
Non-severe defects	1424	118
Severe defects	101	2071

ResNet-101	Non-severe defects	Severe defects
Non-severe defects	1416	126
Severe defects	73	2099

FIGURE 10. Confusion matrix of the proposed method, Bad GAN, ResNet-50, and ResNet-101.

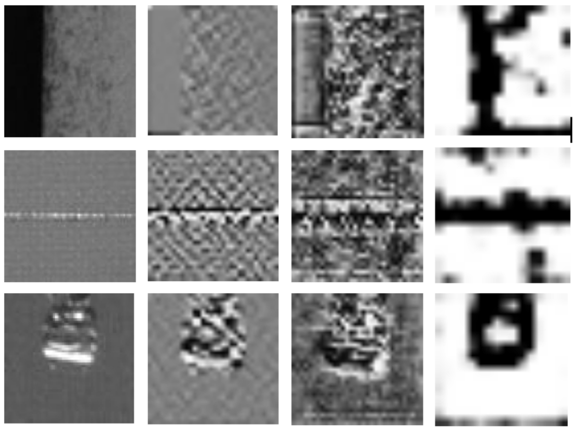


FIGURE 11. Visualization of features in different layers of three severe defective samples from the CCL dataset: the first column is the original image; the second column shows the feature maps from the first convolutional layer (Conv1); the third column illustrates the feature maps from the ninth convolutional layer (Block2); and the last column demonstrates the feature maps from the 17th convolutional layer (Block4).

the convolutional neural network attempts to detect high-level features that are useful for final defect classification as its layers become deeper. It can be clearly seen from the last column of this figure that the network successfully detects the defective patterns of the three image samples, as it only highlights the features of defects in a black and white contrast mode and ignores other unimportant features.

D. COMPUTATIONAL TIME

The convergence curves of our proposed method on the DAGM, NEU and CCL datasets are illustrated in Figure 12. The training loss is descending quickly and smoothly, therefore it does not take many iteration epochs to converge. In addition, the convergence curves of the testing set fit very well to that of the corresponding training set, demonstrating that overfitting does not occur for our proposed method in the three datasets.

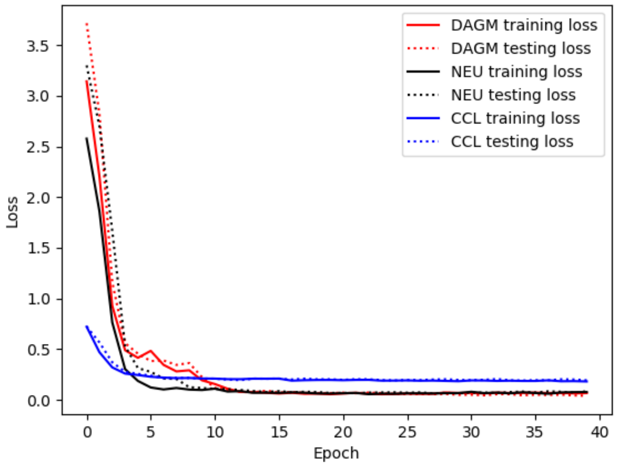


FIGURE 12. Convergence curves of our proposed method for the DAGM, NEU and CCL datasets during training iterations and testing iterations.

The average computational time for detecting an image from the DAGM, NEU and CCL testing sets is demonstrated in Figure 13, shown in milliseconds. For the DAGM testing set, the average computational time per image of our proposed method is 44 milliseconds, while that for the NEU testing set is 10 milliseconds. For the CCL dataset, the computational time per image of our proposed method is 7 milliseconds, while that of Bad GAN, ResNet-50 and ResNet-101 is 2, 14 and 25 milliseconds, respectively. Our proposed method is efficient enough for CCL online defect detection. Its detection speed is slower than Bad GAN, but much faster than ResNet-50 and ResNet-101.

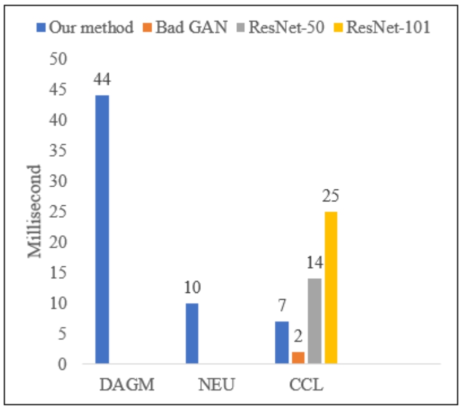


FIGURE 13. Computational time of our proposed method on the DAGM, NEU and CCL datasets as well as the computation time of Bad GAN, ResNet-50 and ResNet-101 in the CCL dataset.

Our proposed method has different performance on the CCL, NEU and DAGM datasets in terms of calculation speed. It achieves the best detection speed on the CCL dataset as it has only 27 convolutional layers and the maximum number of convolution kernels is 128. But for the NEU dataset, it has 35 convolutional layers and the maximum number of convolution kernels is 256. For the DAGM dataset, it has 43 convolutional layers and the maximum number of convolution kernels is 512. Therefore, the speed on the NEU dataset is slightly reduced due to the increase of convolutional layers and kernel numbers. And the speed on the DAGM dataset is greatly reduced, as the convolution kernel number for DAGM is four times that for CCL. This shows that in addition to the increase of the number of convolutional layers, the increase of the number of convolution kernels has a greater impact on the overall calculation speed.

VI. CONCLUSION

In this paper, a generic semi-supervised deep learning approach that requires a small quantity of labeled data for automated surface defect inspection is proposed. While following the MixMatch rules to conduct semi-supervised learning, the proposed method introduces a new method of loss function calculation, employs Cutout technology for data augmentation, and proposes a new convolutional neural network based on residual network structure to achieve accurate ASI. An experiment on two public datasets and one industrial dataset is carried out. The proposed method achieves the best performance in comparisons with several of the best benchmarks in the literature or with the benchmark deep learning methods. In addition, a comparative experiment of model performance given a different number of labeled samples has been conducted, demonstrating that the proposed method can achieve good performance with few labeled training samples. Future work will focus on further improvement of the computational efficiency and stability of ASI semi-supervised learning methods.

ACKNOWLEDGMENT

The authors would like to thank the developers of MixMatch, Cutout, and ResNet.

REFERENCES

[1] C. Steger, M. Ulrich, and C. Wiedemann, *Machine Vision Algorithms and Applications*, 2nd ed. Hoboken, NJ, USA: Wiley, 2018.

[2] A. Hornberg, *Handbook of Machine and Computer Vision: The Guide for Developers and Users*, 2nd ed. Hoboken, NJ, USA: Wiley, 2017.

[3] T. Ozseven, "Surface defect detection and quantification with image processing methods," in *Theoretical Investigations and Applied Studies in Engineering*, T. Özseven Ed. Bursa, Turkey: Ekin Publishing House, 2019, ch. 3, pp. 63–98.

[4] M. Li, S. Wan, Z. Deng, and Y. Wang, "Fabric defect detection based on saliency histogram features," *Comput. Intell.*, vol. 35, no. 3, pp. 517–534, Aug. 2019, doi: 10.1111/coin.12206.

[5] M. W. Ashour, F. Khalid, A. A. Halin, L. N. Abdullah, and S. H. Darwish, "Surface defects classification of hot-rolled steel strips using multi-directional shearlet features," *Arabian J. Sci. Eng.*, vol. 4, pp. 2925–2932, Jun. 2018.

[6] Q. Luo, X. Fang, Y. Sun, L. Liu, J. Ai, C. Yang, and O. Simpson, "Surface defect classification for hot-rolled steel strips by selectively dominant local binary patterns," *IEEE Access*, vol. 7, pp. 23488–23499, 2019.

[7] A. S. Malek, J.-Y. Drean, L. Bigue, and J.-F. Osselin, "Optimization of automated online fabric inspection by fast Fourier transform (FFT) and cross-correlation," *Textile Res. J.*, vol. 83, no. 3, pp. 256–268, Feb. 2013, doi: 10.1177/0040517512458340.

[8] Z. Wen, J. Cao, X. Liu, and S. Ying, "Fabric defects detection using adaptive wavelets," *Int. J. Clothing Sci. Technol.*, vol. 26, no. 3, pp. 202–211, May 2014, doi: 10.1108/Ijcst-03-2013-0031.

[9] J. Ma, Y. Wang, C. Shi, and C. Lu, "Fast surface defect detection using improved Gabor filters," in *Proc. 25th IEEE Int. Conf. Image Process.*, Oct. 2018, pp. 1508–1512.

[10] L. Xu and Q. Huang, "Modeling the interactions among neighboring nanostructures for local feature characterization and defect detection," *IEEE Trans. Autom. Sci. Eng.*, vol. 9, no. 4, pp. 745–754, Oct. 2012, doi: 10.1109/Tase.2012.2209417.

[11] R. Kulkarni, E. Banoth, and P. Pal, "Automated surface feature detection using fringe projection: An autoregressive modeling-based approach," *Opt. Lasers Eng.*, vol. 121, pp. 506–511, Oct. 2019, doi: 10.1016/j.optlaseng.2019.05.014.

[12] X. Xie and M. Mirmehdi, "TEXEMS: Texture exemplars for defect detection on random textured surfaces," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 8, pp. 1454–1464, Aug. 2007.

[13] Z. Zhang, X. Wang, S. Liu, L. Sun, L. Chen, and Y. Guo, "An automatic recognition method for PCB visual defects," in *Proc. Int. Conf. Sens., Diag., Prognostics, Control (SDPC)*, Aug. 2018, pp. 138–142, doi: 10.1109/Sdpc.2018.00034.

[14] L.-F. Chen, C.-T. Su, and M.-H. Chen, "A neural-network approach for defect recognition in TFT-LCD photolithography process," *IEEE Trans. Electron. Packag. Manuf.*, vol. 32, no. 1, pp. 1–8, Jan. 2009.

[15] V. H. Nguyen, V. H. Pham, X. Cui, M. Ma, and H. Kim, "Design and evaluation of features and classifiers for OLED panel defect recognition in machine vision," *J. Inf. Telecommun.*, vol. 1, no. 4, pp. 334–350, Oct. 2017, doi: 10.1080/24751839.2017.1355717.

[16] H. I. Çelik, L. C. Dülger, and M. Topalbekiroğlu, "Development of a machine vision system: Real-time fabric defect detection and classification with neural networks," *J. Textile Inst.*, vol. 105, no. 6, pp. 575–585, Jun. 2014.

[17] R. Ren, T. Hung, and K. C. Tan, "A generic deep-learning-based approach for automated surface inspection," *IEEE Trans. Cybern.*, vol. 48, no. 3, pp. 929–940, Mar. 2018, doi: 10.1109/Tcyb.2017.2668395.

[18] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.

[19] J. Kim, S. Kim, N. Kwon, H. Kang, Y. Kim, and C. Lee, "Deep learning based automatic defect classification in through-silicon via process: FA: Factory automation," in *Proc. 29th Annu. SEMI Adv. Semiconductor Manuf. Conf. (ASMC)*, Saratoga Springs, NY, USA, Apr. 2018, pp. 35–39, doi: 10.1109/ASMC.2018.8373144.

[20] J.-K. Park, B.-K. Kwon, J.-H. Park, and D.-J. Kang, "Machine learning-based imaging system for surface defect inspection," *Int. J. Precis. Eng. Manuf.-Green Technol.*, vol. 3, no. 3, pp. 303–310, Jul. 2016.

[21] Y. Li, D. Zhang, and D.-J. Lee, "Automatic fabric defect detection with a wide-and-compact network," *Neurocomputing*, vol. 329, pp. 329–338, Feb. 2019, doi: 10.1016/j.neucom.2018.10.070.

[22] T. Nakazawa and D. V. Kulkarni, "Wafer map defect pattern classification and image retrieval using convolutional neural network," *IEEE Trans. Semicond. Manuf.*, vol. 31, no. 2, pp. 309–314, May 2018, doi: 10.1109/TSM.2018.2795466.

[23] P. R. Jeyaraj and E. R. Samuel Nadar, "Computer vision for automatic detection and classification of fabric defect employing deep learning algorithm," *Int. J. Clothing Sci. Technol.*, vol. 31, no. 4, pp. 510–521, Aug. 2019, doi: 10.1108/IJCST-11-2018-0135.

[24] Z. Liu, X. Wang, and X. Chen, "Inception dual network for steel strip defect detection," in *Proc. IEEE 16th Int. Conf. Netw., Sens. Control (ICNSC)*. Banff, AB, Canada: IEEE, May 2019, pp. 409–414, doi: 10.1109/ICNSC.2019.8743190.

[25] F. A. Saiz, I. Serrano, I. Barandiarán, and J. R. Sánchez, "A robust and fast deep learning-based method for defect classification in steel surfaces," in *Proc. 9th Int. Conf. Intell. Syst. (IS)*, Funchal-Madeira, Portugal, Sep. 2018, pp. 455–460, doi: 10.1109/IS.2018.8710501.

[26] L. Zhang, Y. Jin, X. Yang, X. Li, X. Duan, Y. Sun, and H. Liu, "Convolutional neural network-based multi-label classification of PCB defects," *J. Eng.*, vol. 2018, no. 16, pp. 1612–1616, Nov. 2018, doi: 10.1049/joe.2018.8279.

[27] D. Soukup and R. Huber-Mörk, "Convolutional neural networks for steel surface defect detection from photometric stereo images," in *Advances in Visual Computing. ISVC* (Lecture Notes in Computer Science), vol. 8887, G. Bebis *et al.*, Eds. Cham, Switzerland: Springer, 2014, pp. 668–677, doi: 10.1007/978-3-319-14249-4_64.

[28] S. Mei, Y. Wang, and G. Wen, "Automatic fabric defect detection with a multi-scale convolutional denoising autoencoder network model," *Sensors*, vol. 18, no. 4, p. 1064, Apr. 2018, doi: 10.3390/s18041064.

[29] A. Mujeeb, W. Dai, M. Erdt, and A. Sourin, "Unsupervised surface defect detection using deep autoencoders and data augmentation," in *Proc. Int. Conf. Cyberworlds (CW)*, Oct. 2018, pp. 391–398, doi: 10.1109/Cw.2018.00076.

[30] Y. Li, W. Zhao, and J. Pan, ''Deformable patterned fabric defect detection with Fisher criterion-based deep learning,'' *IEEE Trans. Autom. Sci. Eng.*, vol. 14, no. 2, pp. 1256–1264, Apr. 2017, doi: 10.1109/Tase.2016.2520955.

[31] W. Zhai, J. Zhu, Y. Cao, and Z. Wang, ''A generative adversarial network based framework for unsupervised visual surface inspection,'' in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2018, pp. 1283–1287.

[32] H. Di, X. Ke, Z. Peng, and Z. Dongdong, ''Surface defect classification of steels with a new semi-supervised learning method,'' *Opt. Lasers Eng.*, vol. 117, pp. 40–48, Jun. 2019.

[33] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. Raffel, ''MixMatch: A holistic approach to semi-supervised learning,'' 2019, *arXiv:1905.02249*. [Online]. Available: http://arxiv.org/abs/1905.02249

[34] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, ''Mixup: Beyond empirical risk minimization,'' 2017, *arXiv:1710.09412*. [Online]. Available: http://arxiv.org/abs/1710.09412

[35] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, ''AutoAugment: Learning augmentation policies from data,'' 2018, *arXiv:1805.09501*. [Online]. Available: http://arxiv.org/abs/1805.09501

[36] T. DeVries and G. W. Taylor, ''Improved regularization of convolutional neural networks with cutout,'' 2017, *arXiv:1708.04552*. [Online]. Available: http://arxiv.org/abs/1708.04552

[37] K. He, X. Zhang, S. Ren, and J. Sun, ''Deep residual learning for image recognition,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778, doi: 10.1109/Cvpr.2016.90.

[38] *DAGM Texture Dataset*. Accessed: Oct. 2019. [Online]. Available: https://hci.iwr.uni-heidelberg.de/node/3616

[39] T. Wang, Y. Chen, M. Qiao, and H. Snoussi, ''A fast and robust convolutional neural network-based defect detection model in product quality control,'' *Int. J. Adv. Manuf. Technol.*, vol. 94, nos. 9–12, pp. 3465–3471, Feb. 2018, doi: 10.1007/s00170-017-0882-0.

[40] K. Song and Y. Yan, ''A noise robust method based on completed local binary patterns for hot-rolled steel strip surface defects,'' *Appl. Surf. Sci.*, vol. 285, pp. 858–864, Nov. 2013, doi: 10.1016/j.apsusc.2013.09.002.

[41] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, ''On large-batch training for deep learning: Generalization gap and sharp minima,'' in *Proc. ICLR*, 2017. [Online]. Available: https://arxiv.org/abs/1609.04836

[42] X. Glorot and Y. Bengio, ''Understanding the difficulty of training deep feedforward neural networks,'' *J. Mach. Learn. Res.*, vol. 9, pp. 249–256, May 2010.

[43] D. Weimer, B. Scholz-Reiter, and M. Shpitalni, ''Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection,'' *CIRP Ann.*, vol. 65, no. 1, pp. 417–420, 2016, doi: 10.1016/j.cirp.2016.04.072.

[44] B. Scholz-Reiter, D. Weimer, and H. Thamer, ''Automated surface inspection of cold-formed micro-parts,'' *CIRP Ann.*, vol. 61, no. 1, pp. 531–534, 2012.

[45] F. Timm and E. Barth, ''Non-parametric texture defect detection using Weibull features,'' *Proc. SPIE*, vol. 7877, Feb. 2011, Art. no. 78770J, doi: Artn.78770j10.1117/12.872463.

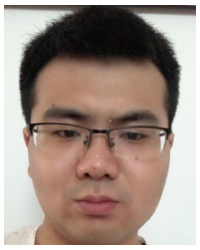
[46] Q. Luo, Y. Sun, P. Li, O. Simpson, L. Tian, and Y. He, ''Generalized completed local binary patterns for time-efficient steel surface defect classification,'' *IEEE Trans. Instrum. Meas.*, vol. 68, no. 3, pp. 667–679, Mar. 2019.

[47] L. Yi, G. Y. Li, and M. M. Jiang, ''An end-to-end steel strip surface defects recognition system based on convolutional neural networks,'' *Steel Res. Int.*, vol. 88, no. 2, pp. 176–187, 2017, doi: ARTN.160006810.1002/srin.201600068.

[48] Y. Gao, L. Gao, X. Li, and X. Yan, ''A semi-supervised convolutional neural network-based method for steel surface defect recognition,'' *Robot. Comput.-Integr. Manuf.*, vol. 61, Feb. 2020, Art. no. 101825.

[49] Z. Dai, Z. Yang, F. Yang, W. W. Cohen, and R. Salakhutdinov, ''Good semi-supervised learning that requires a bad GAN,'' 2017, *arXiv:1705.09783*. [Online]. Available: http://arxiv.org/abs/1705.09783

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 at Honeywell (China) Company, Ltd. Since November 2008, she has been an Assistant Researcher with the Automation College, Hangzhou Dianzi University. Her research interests include industrial process modeling and optimization, machine vision-based surface inspection, and deep learning technology.



HONGCHENG WANG was born in Bozhou, Anhui, China, in 1993. He received the B.S. degree in automation major from the Anhui University of Technology, in 2018. From 2018 to 2019, he was a Graduate Student with the School of Automation, Hangzhou Dianzi University. His research interest includes semi-supervised learning for industrial application.

JIE CHEN was born in Changzhou, Jiangsu, China, in 1996. He received the B.S. degree in electrical engineering and automation from the Jiangsu University of Science and Technology, Jiangsu, in 2018. He is currently pursuing the master's degree with the School of Automation, Hangzhou Dianzi University, Zhejiang, China. His research interests include computer vision and deep learning application.

YAGUANG KONG (Member, IEEE) received the B.Sc. and Ph.D. degrees from Zhejiang University, in 1997 and 2002, respectively. He is currently an Associate Professor with the School of Automation, Hangzhou Dianzi University. His main research interests include computer vision, deep learning, robot visual SLAM, and process automation.

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 the Automation College, Hangzhou Dianzi University. His research interests include industrial automation, modeling, and optimization.