IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 4, 2021, accepted September 6, 2021, date of publication September 9, 2021, date of current version November 8, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3111410

Effective Defect Detection Method Based on Bilinear Texture Features for LGPs

LIBIN HONG, XIANGLEI WU, DIBIN ZHOU, AND FUCHANG LIU
School of Information Science and Technology, Hangzhou Normal University, Hangzhou 311121, China

Corresponding author: Fuchang Liu (liufc@hznu.edu.cn)

This work was supported in part by the Zhejiang Provincial Natural Science Foundation of China under Grant LY20F020017, and in part by the Entrepreneurship and Innovation Project of Hangzhou High Level Returned Talents in 2017.

ABSTRACT Automatic defect detection of light guide plates (LGPs) is an important task in the manufacture of liquid crystal displays. During thermo-printing, defects of tag lines on LGPs may occur easily, and these defects are of two categories: bubbles and missing tag lines. These defects lack salient visual attributes, such as edge-based and region-based features, and as such, traditional methods fail to detect them. To address this, we propose a Dense-bilinear convolutional neural network (BCNN), an end-to-end defect detection network, utilizing Dense-blocks (Huang *et al.*, 2017), Bilinear feature layers (Lin *et al.*, 2015), and squeeze-and-excitation blocks (Hu *et al.*, 2018). Our network exploits fine-grained texture features, which leads to parameter reduction and accuracy enhancement. We validate our network on our LGP dataset containing 5,860 images from three cases: bubbles, tag line existence, and tag line missing. Our network outperforms AlexNet (Krizhevsky *et al.*, 2012), VGG (Simonyan and Zisserman, 2014) and ResNet (He *et al.*, 2016), on both the public and our LGP datasets with less GPU memory consumption.

INDEX TERMS Defects detection, texture classification, bilinear convolutional neural networks.

I. INTRODUCTION

In recent years, liquid crystal displays have become increasingly thinner, and owing to this, the demand for high quality light guide plates (LGPs), which are core components of the backlight module, has increased. Defect detection of LGPs is an essential requirement in liquid crystal displays and can be performed using machine vision. Specifically, the task of LGP defect detection is categorized into three types, bubbles (i.e., defects), tag lines missing (i.e., defects) and tag lines existence (i.e., Normal), as shown in Fig. 1. However, to save cost, industrial detection systems are usually equipped with low-cost cameras and cheap GPUs with small memory sizes. Therefore, defect detection is mostly performed using gray-level cameras and simple image processing algorithms.

Traditional detection methods usually involve image preprocessing for the extraction of edges or regions of LGPs for locating the tag line. However, LGPs have good light transmittance, which causes the image edges to blur and the regions to become inconsistent. This renders the traditional image preprocessing and defect detection algorithms ineffective. As shown in Fig. 2(a), we test the Canny edge detector [7], Gabor filters [8] and OTSU [9] and Partial adaptive threshold methods [10] using LGP images. As can be seen, it is difficult to distinguish the line with no lines and bubbles based on the results of these methods. Fig. 2(b) illustrates the result of tag line detection based on high-order polynomial coefficient line fitting and Gaussian elimination methods [11]. From Fig. 2, we can find that traditional image preprocessing methods and line detection methods are sensitive to blurred edges and inconsistent regions of LGPs. Moreover, detecting bubbles using these methods is challenging. Traditional machine-vision methods are unable to ensure such flexibility as features must be hand-crafted to suit the particular domain. Thus, traditional machine-vision methods conflict the trend moving towards generalization of the production line. Deep learning-based methods provide flexible solutions that can be quickly adapted to new types of products, only using the appropriate number of training images [12]

To address the challenges associated with the defect detection of LGPs using the traditional methods, we herein use texture features that provide rich information for defect classification. Our detection method is mainly based on image texture classification, where the texture features provide the context about the image for inference, and often,

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

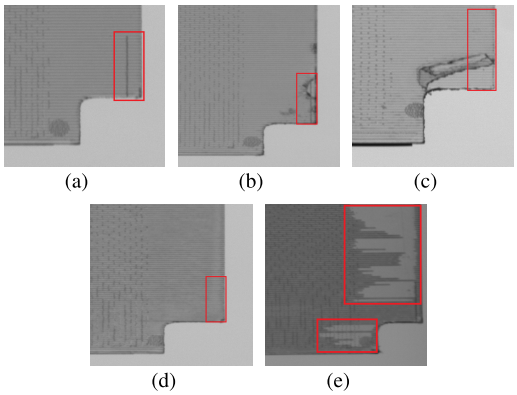
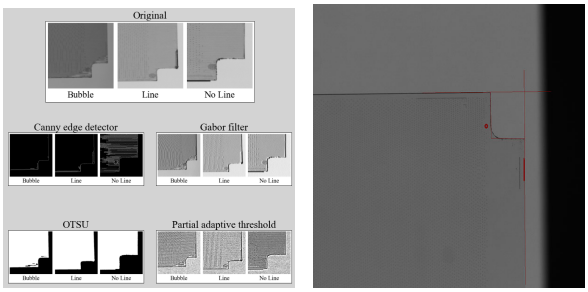


FIGURE 1. Detection of tag line existence (called "NG-0" below), tag line missing (called "NG-1" below), and bubbles (called "NG-2" below). (a) Line without pollution, i.e., defect-free (called "OK" below). (b) Line with pollution at different levels, i.e., "NG-0." (c) and (d) Line missing, i.e., "NG-1." (e) Bubble defect, i.e., "NG-2."



(a) Results of traditional image preprocessing algorithm. (b) Results of detection algorithm based on line fitting.

FIGURE 2. Failures of traditional methods. Red lines in (b) are detected as tag lines, which apparently deviate the true tag line.

the richer the features, the better is the inference. Several studies have focused on designing an optimal filter to extract texture features with high discriminability. Sophisticated hand-designed features cannot be automatically and directly extracted from large datasets. The recent impressive results of deep learning-based methods in machine vision applications have opened up new possibilities for the research and industrial communities. This success can be attributed to the fact that these methods can learn data-driven features, and as such, hand-craft features are not required in such methods. Moreover, deep neural networks are trained end-to-end directly on raw image pixel values. Recently, bilinear convolutional neural networks (BCNNs [2]) were proposed to build orderless texture representations and can be trained in an end-to-end manner. The original BCNNs are based on the VGG [5] backbone. Essentially, the bilinear feature generated by BCNNs can be considered equivalent to the Gram matrix representation, which is a well-known classical two-order texture descriptor. Inspired by BCNNs and to ensure cost-effectiveness (i.e. GPU memory size is a dominant factor of GPU price), we improve BCNNs by reducing the number of parameters achieved by a hybrid framework of BCNNs and Dense-blocks, and boosting the performance of the network with the use of squeeze-and-excitation (SE) blocks.

The main contributions of this work are two-fold:

- We perform defect detection of LGPs using improved BCNNs by replacing the VGG backbone with Dense-blocks and SE-blocks. To the best of our knowledge, few studies have proposed the use of bilinear features for defect detection with potential for practical application. Our method outperforms the state-of-the-art CNNs on our LGP dataset.
- We build an LGP dataset, which is a special type of dataset that we use for defect detection. To the best of our knowledge, there is no publicly available dataset for the automatic defect detection of LGPs. We annotated 5,860 images, based on the three main categories of LGP defects.

Our method only requires gray-level images as input and very few parameters, which significantly increases its applicability in the industry as cheap GPUs can be used. In the remainder of the paper, we first discuss related work, then introduce our method, and finally evaluate and compare it with the state-of-the-art methods.

II. RELATED WORK

Studies concerning defect detection for industrial inspection are scarce. We now review the relevant literature.

A. HAND-CRAFTED FEATURES METHODS

Traditional defect inspection methods usually involve image acquisition, image preprocessing, defect region segmentation, feature extraction, and defect classification. Among these steps, defect region segmentation and feature extraction are the most critical in defect classification. Bi *et al.* [13], Gan and Zhao [14] used region-based and modified region-based segmentation methods to detect Mura defects. Li and Tsai [15] proposed a hough transform-based line detection to identify low-contrast defects in unevenly illuminated images. Lu and Tsai [16] detected scratch and fingerprint defects based on a global image reconstruction scheme using the singular value decomposition method. However, as the aforementioned methods are sensitive to noise and uneven background, they are incapable of detecting various types of defects. Texture-based models are more robust and natural to defect classification. It is generally agreed that the extraction of powerful texture features lead to reliable classification results. The study of texture analysis can be traced back to the earliest work of Julesz [17], who suggested that texture can be modeled using k^{th} order statistics of pixels, also called the cooccurrence statistics. Gray level cooccurrence matrix (GLCM) [18] method was developed based on the cooccurrence statistics. Jiang *et al.* [19] performed weld defect classification using GLCM. Approaches using filters such as the Gabor filter were widely used for texture representation in the early years [8]. Li *et al.* [20] analyzed the texture information of woven fabric using Gabor filters.

However, in a traditional machine-vision fashion, features have to be hand-crafted to suit the particular application. In the final stage, a decision is then made using a hand-crafted rule-based approach or using learning-based classifiers such as support vector machines (SVMs), decision trees, or kNN. Since such classifiers are less powerful than deep-learning methods, the hand-crafted features become very important. In the past, much effort has been made to extract optimal features manually.

B. CNN-BASED METHODS

With the advent of neural networks, it is now possible to perform defect inspection using convolutional neural network (CNN) based methods. CNNs improve flexibility of feature extraction since they are data-driven. The developed methods can be quickly adapted to new types of products and defects by only changing training images. Li [21] used neural networks to learn and detect different types of Mura defects. However, his method requires that the fringe images be enhanced before being learned by neural networks. In fact, popular generic CNN Models can serve as good choices for feature extraction, including AlexNet [4], VGG [5], GoogleNet [22], ResNet [6] and DenseNet [1]. AlexNet was the first to be proposed for this purpose, and the ones proposed later on are deeper and more complex than AlexNet. Deep-learning methods began being applied more often to defect classification problems shortly after the introduction of AlexNet. Masci *et al.* [23] showed their deep-learning approach based surface-defect classification can outperform classic machine-vision approaches where hand-engineered features are combined with SVMs. They achieved excellent results; however, their work was limited to a shallow network (a CNN with five layers), as they did not use ReLU and batch normalization. Faghih-Roohi *et al.* [24] used a similar architecture for the detection of rail-surface defects. They used ReLU activation function and evaluated several networks for the problem of classifying rail defects. Weimer *et al.* [25] evaluated several deep-learning architectures with varying depths of layers for surface-anomaly detection. Some surface-anomaly detections can be addressed as binary-image-classification problems. Therefore, DeepLabv3+ [26] and U-Net [27], normally used for the semantic segmentation, are also applied to defect detection task. However, some defect detections are difficult to recognize using semantic segmentation methods. Recently, Gatys *et al.* [28] showed that the Gram matrix representations extracted from various layers of a VGG can be inverted for texture synthesis. BCNNs [2] yield a pooled outer product of features from two CNNs, identical to Gram matrix representations, i.e., in the 2^{nd} order statistics of pixels. The bilinear pooling of CNN features was proved to be advantageous for texture recognition by Lin and Maji [29].

We reformulate our defect detection into a fine-grained texture recognition problem. Fine-grained texture recognition is a challenging problem and has recently emerged as an active topic, due to the diverse appearance and complex structure of texture, high intra-class variability and small inter-class differences. Similar to the traditional texture methods, the bilinear feature vector is an orderless representation of the input image and is therefore suitable for modeling textures. Compared to the related methods, the approach proposed in this paper follows an end-to-end design with the DenseNet network and the Bilinear network. Our method is evaluated and compared with the state-of-the-art methods. The results provide an insight in the complexity of different industrial defects recognition tasks.

III. METHOD

A. OVERVIEW OF THE NETWORK

As discussed in Section I, GPU cost and performance are critical aspects that need to be considered for industrial defect detection. A drawback of the bilinear features is the memory overhead of storing the high-dimensional features. Thus, our method is based on DenseNet, which requires less parameters and does not suffer from the problem of gradient vanishing, thereby meeting the economic requirement. For detection accuracy, we use BCNNs to extract orderless texture features and augment the networks using SE-blocks that strengthen their representational power by adaptively recalibrating channel-wise feature responses. Therefore, our proposed network meets the requirements of cost and performance for industrial applications.

Fig. 3 illustrates the architecture of our network for LGP defect detection. The design of the defect classification network follows two important principles. First, the appropriate capacity for large complex appearances is ensured by using several layers of convolution and bilinear modules. This enables the network to capture not only the local texture features, but also the global ones that span a large area of the image. We also consider features between channels and use SE-blocks to improve the quality of representations produced by the network by explicitly modeling the interdependencies between the channels of its convolutional features. Second, our network should reduce the overfitting to the large number of parameters. We employ Dense-blocks which introduces a shortcut that the network can use to avoid using a large number of feature maps, if they are not needed. In our network, we use five Dense-blocks, four SE-blocks, and a Bilinear feature layer in the end; there are therefore a total of 118 convolution layers. We also employ transition layers; 1×1 conv and 2×2 average pooling by stride 2. The main difference between DenseNet and our network is that in our network, the final feature passes through a pooled outer product, thereby leading to the generation of an orderless texture feature. In the experiments, we find that our architecture achieves higher precision than VGGs based BCNNs.

B. OUR METHODS

1) FEATURE FUNCTIONS AND LOSS

The bilinear feature (or the Gram matrix form) is a type of orderless representation of an image and is therefore a

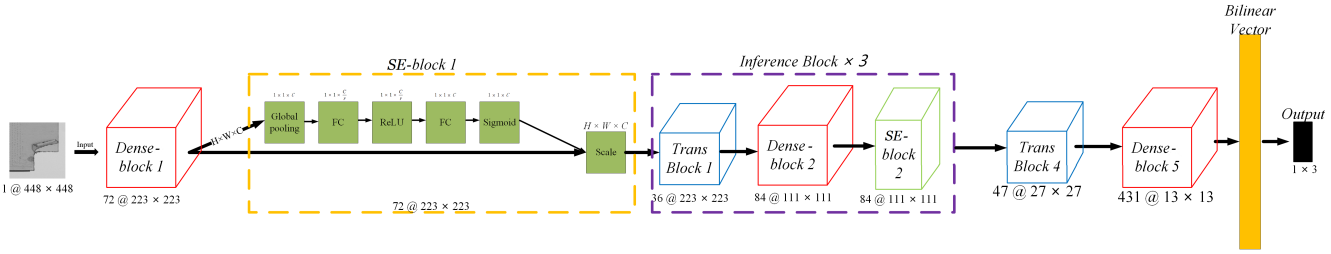


FIGURE 3. Architecture for implementing our approach. It is based on DenseNet backbone, to which SE-blocks and Bilinear feature layers are added. Input is a gray-level image, and output is the class, similar to the bubble defect, tag line existing, and tag line missing.

decent texture descriptor. The simplest bilinear feature layer can be implemented using a pooled outer product of features derived from two identical convolutional features. The bilinear layer is closely related to the Second-Order Pooling approach [30]. However, more studies show that the bilinear feature is generic, and several texture representations can be written as bilinear features. The paper [2] has shown that various orderless texture descriptors can be written in the bilinear form and derive variants that are end-to-end trainable. Moreover, bilinear layers can be easily plugged into existing CNNs or domain-specific fine-tuning for transfer learning. Our loss function is the cross-entropy cost function for classification. The feature outputs are given by the sum of the pooling of the outer products of features from the last Dense-block.

In the classification task, the BCNN model $\mathcal{B}$ is defined as quadruple $\mathcal{B} = (f_A, f_B, \mathcal{P}, \mathcal{C})$, where f_A and f_B are two CNN feature function, $\mathcal{P}$ is a pooling function, and $\mathcal{C}$ is a classification function. This BCNN extracts deep visual ϕ for image $\mathcal{I}$ as below:

$$\phi(I) = \sum_{l \in L} bilinear(l, I, f_A, f_B) \tag{1}$$

where $bilinear(l, I, f_A, f_B) = f_A(l, I^{\tau}) f_B(l, I)$ is the bilinear feature combination of f_A and f_B at each location $l \in \mathcal{L}$. The mapping function $f: \mathcal{I} \times \mathcal{L} \rightarrow \mathcal{R}^{c \times D}$ outputs a feature vector of size $c \times \mathcal{D}$ for image $\mathcal{I}$ at location $\mathcal{L}$. For the classification task, function $\mathcal{C}$ is trained using image features ϕ. Note that ϕ is a high-dimensional feature vector.

In our network, we implement B by concatenating the feature output from $H(\cdot)$ into a vector and feeding it into an outer product. Fig. 4 and Fig. 5 provide the details of implementing B and $H(\cdot)$. Our network comprises 100 convolution layers, 9 connected layers, and 115 ReLUs and batch normalization layers. Table 1 shows that the use of the bilinear feature layer leads to a significant improvement in accuracy and loss.

To exploit the discriminability of BCNNs, we employ the SE-blocks and Dense-blocks. These blocks can improve the quality of representations produced by a network by explicitly modeling the interdependencies between the channels of its convolutional features. Our ablation study demonstrates the effectiveness of using SE-blocks and Dense-blocks for the precision and recall of the network. Dense-blocks help the network converge to a low loss, and SE-blocks can achieve high precision on the validation set.

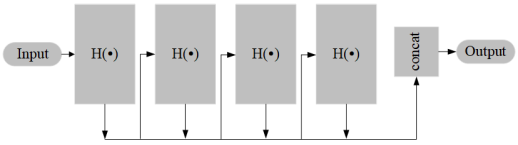


FIGURE 4. Architecture of the Dense-block. Dense-block 1-4 consist of four $H(\cdot)$ functions, while Dense-block 5 has 32 $H(\cdot)$.

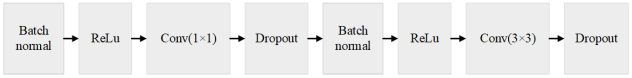
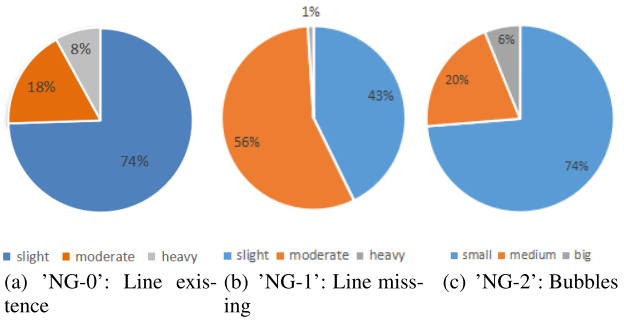


FIGURE 5. Detailed configuration of the $H(\cdot)$ function.



(a) 'NG-0': Line existence (b) 'NG-1': Line missing (c) 'NG-2': Bubbles

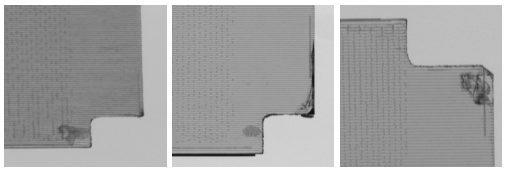
FIGURE 6. Statistics of the LGP dataset: (a) Tag lines existing with different levels of polluted backgrounds. (b) Disappearance of tag lines caused by different levels of print contamination. (c) Defects obtained by varying sizes of bubbles.

2) END-TO-END TRAINING

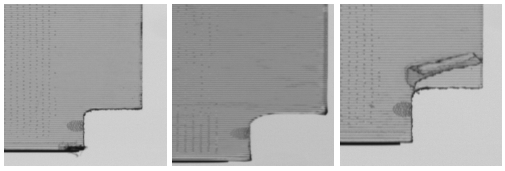
Compared to existing methods, we do not have to transfer weights from pre-trained weights to initialize BCNNs. We directly train our network from scratch in an end-to-end manner; consequently, we find that our training converges fast.

IV. EXPERIMENTS AND ANALYSIS

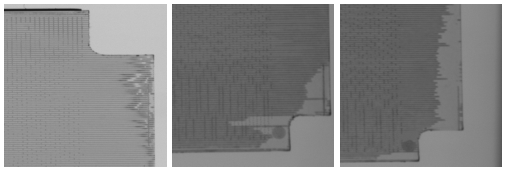
The proposed network is extensively evaluated on the defect detection of LGPs and objects classification. This section first presents the details of the dataset and then presents the details of the evaluation and its results. We implement our end-to-end



(a) 'NG-0': Tag line existing with slight, moderate, heavy blurring

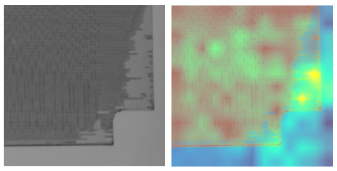


(b) 'NG-1': Tag line missing cased by slight, moderate, heavy contamination

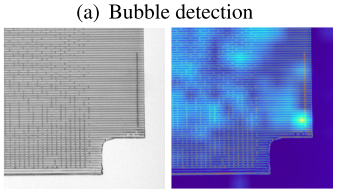


(c) 'NG-2': Bubble with small, medium, big size

FIGURE 7. Instances of the LGP dataset.



(a) Bubble detection



(b) Line detection

FIGURE 8. Visual explanations generated by our method on LGP images with bubbles and the tag line. The heatmaps provide insight into what the network focused on: bubbles or tag lines.

architecture with Tensorflow and run it on an i5-10400F CPU with an NVIDIA Geforce RTX3060 12GB.

A. LGP DATASETS

The LGP dataset comprises 4 categories and 3,969 gray-level images, each of 448 × 448 pixels and captured using industrial CMOS cameras. Note that we perform data augmentation on categories of "NG-0," "NG-1," and "NG-2" by flipping images horizontally and vertically. Finally, we obtain 6,905 images, comprising 1,419 images of "NG-0," 889 images of "NG-1," 2,097 images of "NG-2," and 2,500 images of "OK." Fig. 7 presents instances of different types in the LGP dataset. The corresponding statistics of the LGP dataset are shown in Fig. 6. We split the LGP dataset into a train set (5,594 images), validation set (620 images) and test set (691 images). Fig. 6 illustrates the distribution of the bubble sizes, tag line pollution levels, and contamination levels. Small bubbles, heavy pollution, and heavy contamination are also identified, and they occupy a certain portion; these can be regarded as hard samples.

B. OPTIMIZATION USING RECTIFIED ADAM

Now, an adaptive learning rate is used to accelerate the optimization of the deep learning model, which is the main method for developing the optimizer. Many optimization methods have been proposed, such as the adaptive gradient algorithm, Adadelta, Adamax, root mean square propagation, adaptive moment optimization (Adam), or Nesterov adaptive moment optimization. Rectified Adam is one of the most progressive algorithms, which was developed by [31]. It improves the generalization, and introduces a term to rectify the variance of the adaptive learning rate by applying warm up with a low initial learning rate. Rectified Adam has been confirmed in project research and achieved success [32].

Computing the weights according to the Adam optimizer:

$$W_t = W_{t-1} - \eta \frac{\hat{m}^t}{\sqrt{\hat{\upsilon}} + \xi} \tag{2}$$

The first moving momentum:

$$m_t = (1 - \beta_1) \sum_{i=0}^{t} \beta_1^{t-i} g_i \tag{3}$$

The second moving momentum:

$$\upsilon_t = (1 - \beta_2) \sum_{i=0}^{t} \beta_2^{t-i} g_i^2 \tag{4}$$

The bias correction of the momentums:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{5}$$

$$\hat{\upsilon}_t = \frac{\upsilon_t}{1 - \beta_2^t} \tag{6}$$

Adding the rectification term in Equation (1), the recent variant of the Adam optimization, named rectified Adam (RAdam), has the form:

$$W_t = W_{t-1} - \eta \frac{\hat{m}^t}{\sqrt{\hat{\upsilon}}} \tag{7}$$

where the step size, η, is an adjustable hyperparameter and rectification rate is:

$$r_t = \sqrt{\frac{(p_t - 4)(p_t - 2)p_\infty}{(p_\infty - 4)(p_\infty - 4)p_t}} \tag{8}$$

with

$$p_t = p_\infty - \frac{2t\beta_2^t}{1 - \beta_2^t} \quad \text{and} \quad p_\infty = \frac{2}{(1 - \beta_2^t)} - 1$$

When the length of the approximated simple moving average is less than or equal to 4, the variance of the adaptive learning rate is deactivated. Otherwise, the variance rectification term is calculated and parameters are updated with the adaptive learning rate.

TABLE 1. Our method vs. popular methods on the LGP without noise using 10-fold cross-validation.

Indication	Accuracy	Recall	Precision	STD
Alexnet	84.34%	79.21%	81.77%	1.265
ResNet101	83.32%	79.08%	81.20%	3.053
MobileNet	31.41%	34.11%	43.41%	6.014
ShuffleNet	24.98%	30.37%	45.97%	5.433
DenseNet121	85.67%	82.41%	85.02%	1.529
VGG16	86.76%	83.04%	84.63%	1.394
BCNN	89.04%	88.62%	93.09%	0.876
BCNN+SE	91.03%	90.02%	93.81%	**0.603**
BCNN+Dense	93.48%	**91.94%**	**96.90%**	1.037
BCNN+SE+Dense	**93.52%**	**91.94%**	96.45%	0.625

TABLE 2. Our method vs. popular methods on the LGP with Gaussian noise ($\sigma = 0.1$) using 10-fold cross-validation.

Noise	Accuracy	Recall	Precision	STD
Alexnet	73.60%	65.58%	65.80%	3.876
ResNet101	48.03%	47.17%	57.11	12.373
MobileNet	25.96%	30.03%	46.24	6.170
ShuffleNet	33.94%	25.89%	15.19%	6.759
DenseNet121	56.37%	53.85%	58.52%	12.741
VGG16	76.97%	69.99%	71.53%	2.390
BCNN	84.32%	80.54%	86.46%	2.770
BCNN+SE	85.63%	81.09%	86.20%	1.470
BCNN+Dense	86.93%	83.34%	88.91%	**1.336**
BCNN+SE+Dense	**87.35%**	**83.76%**	**89.43%**	1.374

TABLE 3. Our method vs. popular methods on the LGP with Gaussian noise ($\sigma = 0.5$) using 10-fold cross-validation.

Noise	Accuracy	Recall	Precision	STD
Alexnet	66.81%	59.33%	58.9%	4.028
ResNet101	27.27%	17.07%	25.25%	6.752
MobileNet	35.15%	31.21%	40.13%	4.305
ShuffleNet	31.53%	26.29%	15.75%	9.601
DenseNet121	28.72%	33.99%	20.53%	6.537
VGG16	50.63%	46.42%	48.39%	6.850
BCNN	84.27%	79.08%	85.3%	1.763
BCNN+SE	84.01%	79.38%	85.03%	1.520
BCNN+Dense	84.40%	79.80%	85.04%	1.747
BCNN+SE+Dense	**84.73%**	**80.41%**	**85.96%**	**1.109**

C. RESULTS ON THE LGP DATASET AND PUBLIC DATASETS

In this section, we evaluate our method first on public datasets and then on the LGP dataset containing images that have Gaussian noise in them. In Table 1, we compare our method with popular classification CNNs, such as AlexNet, VGG16, BCNN based on VGG backbone, ResNet with 101 layers, ShuffleNet (version 2), Mobilenet (version 3), and DenseNet on public datasets. Our method (herein, referred to as "BCNN + SE + Dense," where "SE" and "Dense" means SE-blocks and Dense-blocks, respectively) achieves the best performance on most of the metrics, and outperforms most of the aforementioned methods significantly. Although our method is slightly inferior to BCNN based on DenseNet backbone, the STD of 10-fold in terms of accuracy shows that our method is more stable. Note that accuracy, recall, precision, and STD in Table 1 to Table 7 are the mean of four categories.

To further evaluate our method, we add Gaussian noise on the LGP images and test our network using ResNet and ShuffleNet. Table 2 to Table 7 reports that our network is robust to different levels of noise. Although, in the case of Gaussian noise ranging from $\sigma = 0.1$ to $\sigma = 1.0$, the accuracy of our network drops slightly, our network has smaller STD in terms of accuracy than most of other methods at all levels of noises. Similarly, we achieve good performance in the case of salt and pepper noise ranging from 5% to 20%. Moreover, we find

TABLE 4. Our method vs. popular methods on the LGP with Gaussian noise ($\sigma = 1.0$) using 10-fold cross-validation.

Noise	Accuracy	Recall	Precision	STD
Alexnet	62.93%	55.29%	55.87%	3.340
ResNet101	30.46%	25.10%	10.20%	**0.2855**
MobileNet	32.68%	31.83%	41.43%	4.561
ShuffleNet	29.66%	23.30	12.03	6.889
DenseNet121	28.66%	26.98%	23.84%	7.868
VGG16	42.21%	39.43%	41.93	13.29
BCNN	79.31%	72.74%	78.66%	2.039
BCNN+SE	78.73%	73.29%	77.99%	1.701
BCNN+Dense	**81.16%**	**74.66%**	**81.04%**	1.132
BCNN+SE+Dense	80.82%	74.09%	80.1%	1.376

TABLE 5. Our method vs. popular methods on the LGP with 5% salt and pepper noise using 10-fold cross-validation.

Noise	Accuracy	Recall	Precision	STD
Alexnet	65.40%	59.97%	63.83%	4.847
ResNet101	32.04%	31.97%	36.47%	4.840
MobileNet	29.07%	32.35%	46.27%	4.546
ShuffleNet	39.06%	33.98%	44.81%	7.342
DenseNet121	59.22%	52.41%	59.46%	7.370
VGG16	75.19%	69.77%	70.06%	**1.979**
BCNN	84.70%	83.12%	82.48%	3.924
BCNN+SE	85.38%	80.59%	85.04%	3.613
BCNN+Dense	88.79%	**86.71%**	89.50%	2.198
BCNN+SE+Dense	**89.07%**	86.21%	**91.62%**	2.493

TABLE 6. Our method vs. popular methods on the LGP with 10% salt and pepper noise using 10-fold cross-validation.

Noise	Accuracy	Recall	Precision	STD
Alexnet	66.92%	61.24%	61.39%	2.737
ResNet101	31.20%	29.02%	32.53%	6.926
MobileNet	28.27%	31.79%	46.53%	2.962
ShuffleNet	41.65%	38.24%	44.17%	7.266
DenseNet121	41.99%	39.33%	44.72%	10.39
VGG16	71.38%	65.55%	65.53%	2.332
BCNN	77.42%	74.59%	75.26%	4.710
BCNN+SE	79.10%	75.84%	76.98%	4.156
BCNN+Dense	80.45%	75.01%	78.99%	3.836
BCNN+SE+Dense	**84.13%**	**79.69%**	**83.80%**	**2.050**

TABLE 7. Our method vs. popular methods on the LGP with 20% salt and pepper noise using 10-fold cross-validation.

Noise	Accuracy	Recall	Precision	STD
Alexnet	63.40%	56.8%	57.85%	4.199
ResNet101	32.32%	29.25%	31.62%	3.584
MobileNet	29.15%	31.09%	45.67%	3.279
ShuffleNet	38.63%	34.75%	45.43%	12.68
DenseNet121	27.37%	27.12%	27.89%	8.389
VGG16	**68.74%**	62.44%	**62.78%**	**2.290**
BCNN	56.22%	52.12%	55.87%	8.291
BCNN+SE	58.53%	55.24%	49.17%	7.816
BCNN+Dense	65.35%	59.55%	59.88%	7.349
BCNN+SE+Dense	66.16%	**62.61%**	61.78%	7.768

TABLE 8. Comparisons with popular methods in terms of parameter, training time, and inference time.

Methods	Param(MB)	Time(MS)	Minutes/Epoch
AlexNet	233.08	5.08	0.86
ResNet101	170.55	11.01	1.15
MobileNet	13.37	7.74	0.40
ShuffleNet	8.69	5.66	0.17
DenseNet121	30.44	18.86	3.82
VGG16	527.79	25.11	1.15
BCNN	60.13	12.13	1.10
BCNN+SE	60.14	13.79	1.14
BCNN+Dense	63.28	15.14	1.29
BCNN+SE+Dense	63.29	15.21	1.38

that our method is better in terms of recall. Note that recall is significant in industrial defect detection. We note that Alexnet and VGG16 achieves better performance except in terms of recall for heavy salt and pepper noise. These old networks may shed light on how a network robust against noise attacks can be designed.

Table 8 reports GPU memory usage as well as training and inference time for different methods. We observe that our method provides a good balance between network parameters, training time, and inference time. Our network consumes less GPU memory, and achieves fast inference. However, our training time is slightly longer than that of other methods. Nevertheless, it is acceptable for industrial applications.

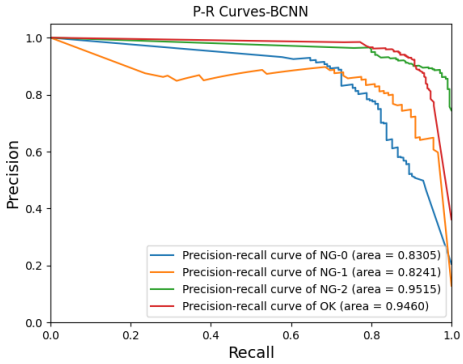
We also provide a visual explanation for the output of our last convolutional layer in Fig. 8. We observe that the position of the highest responses of our network can efficiently locate the defect areas. Our network can capture the features of bubbles and the tag line.

D. ABLATION STUDY

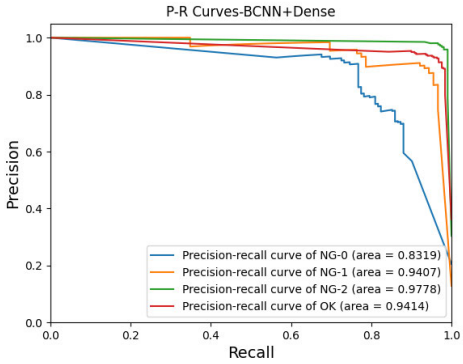
To investigate the behavior of our proposed method, we conduct several ablation studies. First, we investigate the effect of bilinear layers. To do this, we compare DenseNet with BCNN with Dense-blocks; we observe that the latter is superior to DenseNet, as listed in Table 1. To validate the effect of the SE-blocks, we compare BCNN with and without SE-blocks; we find that SE-blocks lead to improvement in most cases, as listed in Table 1. Even for Gaussian and salt and pepper noise, we can still make improvements with SE-blocks, as listed in Table 2 to Table 7. Furthermore, we present the Precision–Recall curves of four alternatives, including BCNN, BCNN with Dense-blocks, BCNN with SE-blocks, and BCNN with SE-blocks and Dense-blocks. As shown in Fig. 9, we observe that SE-blocks and Dense-blocks improve the performance significantly, while achieving higher AUC values. The results on the Precision–Recall curves are consistent with our results in Table 2 to Table 7. Moreover, Fig. 10 and Fig. 11 shows that our network converges faster during training.

E. INDUSTRIAL SCENARIOS TEST

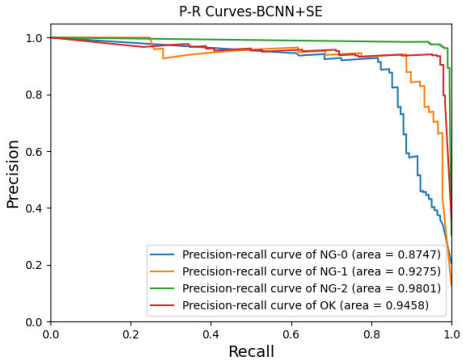
We also conduct a realistic test of our method using LGP images captured from a production line. Fig. 12 illustrates our



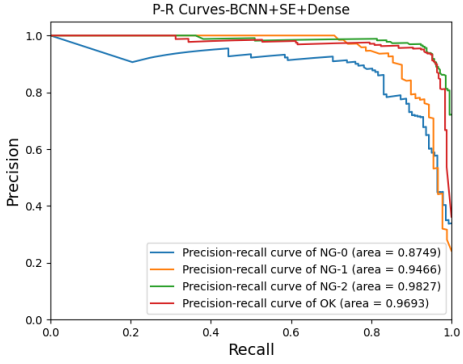
(a) Precision-Recall curve of BCNN



(b) Precision-Recall curve of BCNN+Dense-blocks



(c) Precision-Recall curve of BCNN+SE-blocks



(d) Precision-Recall curve of BCNN+SE-blocks+Dense-blocks

FIGURE 9. Precision-recall curves of our approach and other alternatives. (a), (b), (c) and (d) show precision-recall curves and AUCs. The curve with high AUC value represents its method has good classification performance.

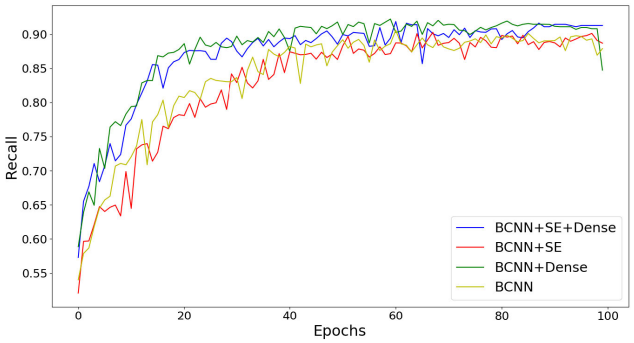


FIGURE 10. Ablation study on the LGP by comparing our approach in terms of recall with two alternatives: DenseNet with and without SE-blocks.

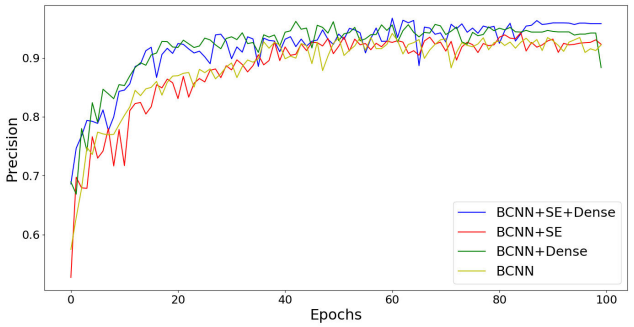


FIGURE 11. Ablation study on the LGP by comparing our approach in terms of precision with two alternatives: DenseNet with and without SE-blocks.

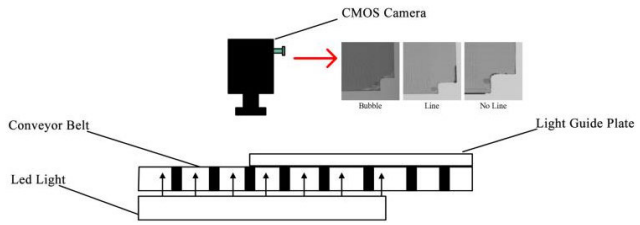


FIGURE 12. Our LGP defect detection system. Industrial CMOS camera is facing the conveyor belt, and it automatically captures the LGP images. Our network is being run on a GPU in the backend.

industrial set up, which includes an industrial CMOS camera and some software modules. The camera captures the LGP images and sends them to our network via backend. Our network runs on a GPU and is capable of processing images with a speed of approximately 66 FPS. Since our network requires an exceptionally low GPU memory footprint (63.29 MB), our method can also be run on a cheap GPU with a small memory, which is advantageous for the industrial community.

We compare our method with the traditional method, our method achieves a high true/false positive, and its performance has also been verified in a real production line. Notably, traditional image processing algorithms are not robust to noise and regional inconsistency as well as not flexible to new task.

We also present the confusion matrix of our method in Fig. 13. Our algorithm yields an extremely high true positive and extremely low false positive and negative. Our

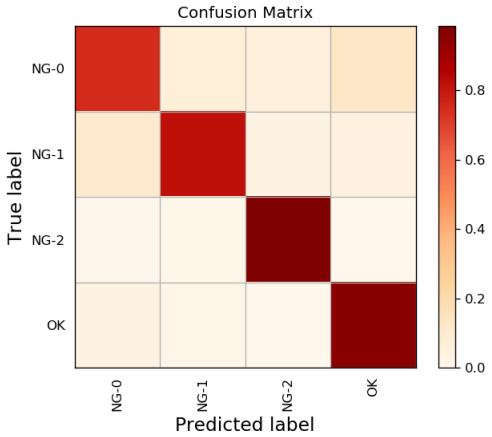


FIGURE 13. Confusion matrix of our algorithm. Yellow indicates 1.0 and purple indicates 0.0.

method demonstrates good discrimination ability and less confusion between different classes.

V. CONCLUSION AND DISCUSSION

In industrial processes, one of the most important tasks is defect detection, which ensures the quality of the finished product. Often, quality control is carried out manually and workers are trained to identify complex defects. Such control is, however, very time consuming, inefficient, and results in a serious limitation of the production capacity.

This paper explored a deep-learning approach to surface-defect detection with a texture classification network from the point of view of specific industrial application. We propose a novel end-to-end LGP defect detection network based on an improved version of BCNNs. Specifically, we aim to detect three types of cases: tag line existence, tag line missing defect, and bubble defect. To address the challenges associated with noise and regional inconsistency in images, we introduce Dense-blocks and SE-blocks to the BCNNs to improve the classification discriminability of defects texture. Our method achieves the best performance compared to AlexNet, VGG, ResNet, ShuffleNet, Mobilenet and DenseNet. Furthermore, our network requires less parameters and is suitable for application in cheap GPUs with less memory for industrial inspection. We also verify our method in real industrial scenarios and confirm that it achieves superior performance for use by the industrial community. In future work, we will focus on acquiring new complex datasets based on real-world visual defects inspection problems, where deep-learning (and other) methods could be realistically evaluated in full extent; the dataset presented in this paper is a first step in this direction. Our CNN model still can be optimized further in terms of inference and training time.

REFERENCES

[1] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2261–2269.

[2] T.-Y. Lin, A. RoyChowdhury, and S. Maji, "Bilinear CNN models for fine-grained visual recognition," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1449–1457.

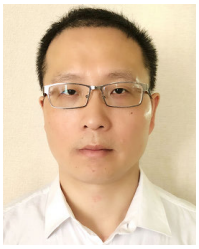
[3] J. Hu, L. Shen, and G. Sun, "Squeeze- and-excitation networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7132–7141.
[4] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Neural Inf. Process. Syst. Conf. (NIPS)*, 2012, pp. 1097–1105.
[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: https://arxiv.org/abs/1409.1556
[6] 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.
[7] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 6, pp. 679–698, Nov. 1986.
[8] B. S. Manjunath and W. Y. Ma, "Texture features for browsing and retrieval of image data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 8, pp. 837–842, Aug. 1996.
[9] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-9, no. 1, pp. 62–66, Jan. 1979.
[10] S. D. Yanowitz and A. M. Bruckstein, "A new method for image segmentation," *Comput. Vis., Graph., Image Process.*, vol. 46, no. 1, pp. 82–95, 1989.
[11] V. Rovenski, *Modeling of Curves and Surfaces With MATLAB*. New York, NY, USA: Springer, 2010.
[12] D. Tabernik, S. Sela, J. Skvarc, and D. Skocaj, "Segmentation-based deep-learning approach for surface-defect detection," *J. Intell. Manuf.*, vol. 31, pp. 759–776, May 2010.
[13] X. Bi, C. Zhuang, and H. Ding, "A new Mura defect inspection way for TFT-LCD using level set method," *IEEE Signal Process. Lett.*, vol. 16, no. 4, pp. 311–314, Apr. 2009.
[14] Y. Gan and Q. Zhao, "An effective defect inspection method for LCD using active contour model," *IEEE Trans. Instrum. Meas.*, vol. 62, no. 9, pp. 2438–2445, Sep. 2013.
[15] W. C. Li and D. M. Tsai, "Defect inspection in low-contrast LCD images using Hough transform-based nonstationary line detection," *IEEE Trans. Ind. Informat.*, vol. 7, no. 1, pp. 136–147, Feb. 2011.
[16] C.-J. Lu and D.-M. Tsai, "Automatic defect inspection for LCDs using singular value decomposition," *Int. J. Adv. Manuf. Technol.*, vol. 25, nos. 1–2, pp. 53–61, Jan. 2005.
[17] B. Julesz, "Visual pattern discrimination," *IRE Trans. Inf. Theory*, vol. 8, no. 2, pp. 84–92, Feb. 1962.
[18] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-3, no. 6, pp. 610–621, Nov. 1973.
[19] H. Jiang, Y. Zhao, J. Gao, and Z. Wang, "Weld defect classification based on texture features and principal component analysis," *Insight, Non-Destructive Test. Condition Monitor.*, vol. 58, no. 4, pp. 194–199, 2016.
[20] L. Q. Li, T. T. Shan, L. Xue, J. Wang, and X. Chen, "Study on woven fabric texture based on Fourier transform and Gabor transform," *Key Eng. Mater.*, vol. 671, pp. 369–377, Nov. 2015.
[21] T. Y. Li, J. Z. Tsai, R. S. Chang, L. W. Ho, and C. F. Yang, "Pretest gap Mura on TFT LCDs using the optical interference pattern sensing method and neural network classification," *IEEE Trans. Ind. Electron.*, vol. 60, no. 9, pp. 3976–3982, Sep. 2013.
[22] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.
[23] J. Masci, U. Meier, D. Ciresan, J. Schmidhuber, and G. Fricout, "Steel defect classification with max-pooling convolutional neural networks," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jun. 2012, pp. 1–6.
[24] S. Faghih-Roohi, S. Hajizadeh, A. Nunez, R. Babuska, and B. De Schutter, "Deep convolutional neural networks for detection of rail surface defects," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2016, pp. 2584–2589.
[25] D. Weimer, B. Scholz-Reiter, and M. Shpitalni, "Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection," *CIRP Ann.-Manuf. Technol.*, vol. 65, no. 1, pp. 417–420, 2016.
[26] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proc. Eur. Conf. Comput. Vis.*, Sep. 2018, pp. 801–818.
[27] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015* (Lecture Notes in Computer Science), vol. 9351. Cham, Switzerland: Springer, 2015.
[28] L. A. Gatys, A. S. Ecker, and M. Bethge, "Texture synthesis using convolutional neural networks," in *Proc. Neural Inf. Process. Syst. Conf. (NIPS)*, 2015, pp. 262–270.
[29] T.-Y. Lin and S. Maji, "Visualizing and understanding deep texture representations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2791–2799.
[30] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu, "Semantic segmentation with second-order pooling," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2012, pp. 430–443.
[31] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the variance of the adaptive learning rate and beyond," 2019, *arXiv:1908.03265*. [Online]. Available: https://arxiv.org/abs/1908.03265
[32] D. O. Melinte and L. Vladareanu, "Facial expressions recognition for human–robot interaction using deep convolutional neural networks with rectified Adam optimizer," *Sensors*, vol. 20, no. 8, p. 2393, 2020.

LIBIN HONG was born in Hangzhou, Zhejiang, China, in 1980. He received the M.Sc. degree in computing and software technology from the University of Wales, Swansea, U.K., in 2005, and the Ph.D. degree in computer science from the University of Nottingham, in 2018.

From 2005 to 2008, he was an Officer at State Street Corporation and was awarded the GIS/Global Integration Group Award, in 2008. He is currently a Lecturer and the Vice Head of the Computer and Financial Information Services Department, School of Information Science and Technology, Hangzhou Normal University. His research interests include evolutionary computation, hyper-heuristics, metaheuristics, and deep learning.

XIANGLEI WU was born in Fuyang, Anhui, China, in 1997. He received the B.S. degree in computer science from Fuyang Normal University. He is currently pursuing the M.S. degree with the School of Information Science and Technology, Hangzhou Normal University.

His research interests include evolutionary computation and deep learning.



DIBIN ZHOU was born in Anyi, Jiangxi, China, in 1978. He received the Ph.D. degree in computer science from Zhejiang University, in 2008.

He was a Research Associate with British Bedfordshire University, U.K. He is currently a Lecturer with the School of Information Science and Technology, Hangzhou Normal University. His research interests include commercial digital image processing software design and development, machine vision, deep learning, and wireless internet planning.

FUCHANG LIU was born in Nanjing, Jiangsu, China, in 1982. He received the B.S. and Ph.D. degrees in computer science from Nanjing University of Science and Technology, in 2004 and 2009, respectively.

He was a Postdoctoral Research Fellow in computer science and engineering with Ewha Woman University and Nanyang Technological University. He is currently an Associate Professor with the School of Information Science and Technology, Hangzhou Normal University. His research interests include scene understanding, computer graphics, GPU rendering, and collision detection.