Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/MBE

MBE, 18(4): 3755–3780.
DOI: 10.3934/mbe.2021189
Received: 11 March 2021
Accepted: 26 April 2021
Published: 29 April 2021

Research article

Defect detection in code characters with complex backgrounds based on BBE

Jianzhong Peng[1,2], Wei Zhu[1,2], Qiaokang Liang[1,2,*], Zhengwei Li[3], Maoying Lu[4], Wei Sun[1,2] and Yaonan Wang[1,2]

[1] College of Electrical and Information Engineering, Hunan University, Changsha 410082, China
[2] National Engineering Laboratory for Robot Vision Perception and Control, Hunan University, Changsha 410082, China
[3] Department of Mechanical Engineering, University of Alberta, Edmonton, Alberta T6G2E1, Canada
[4] T-Line Technology CO., LTD, Suqian 223700, China

* **Correspondence:** Email: qiaokang@mail.ustc.edu.cn; Tel: +86073188822224.

Abstract: Computer vision technologies have been widely implemented in the defect detection. However, most of the existing detection methods generally require images with high quality, and they can only process code characters on simple backgrounds with high contrast. In this paper, a defect detection approach based on deep learning has been proposed to efficiently perform defect detection of code characters on complex backgrounds with a high accuracy. Specifically, image processing algorithms and data enhancement techniques were utilized to generate a large number of defect samples to construct a large data set featuring a balanced positive and negative sample ratio. The object detection network called BBE was build based on the core module of EfficientNet. Experimental results show that the mAP of the model and the accuracy reach 0.9961 and 0.9985, respectively. Individual character detection results were screened by setting relevant quality inspection standards to evaluate the overall quality of the code characters, the results of which have verified the effectiveness of the proposed method for industrial production. Its accuracy and speed are high with high robustness and transferability to other similar defect detection tasks. To the best of our knowledge, this report describes the first time that the BBE has been applied to defect inspections for real plastic container industry.

Keywords: defect detection; deep learning; EfficientNet; BBE; character recognition; transfer learning

1. Introduction

With the coming of the consumption age, many types of product packages have been widely utilized in our daily life. Therefore, the code characters printed on these product packages have attracted mounting attention due to the fact that they carry important information for the consumers, as well as the enterprises. Specifically, if the code characters are not clear or the related information is not complete, it will not only affect the product sales, but also damage the reputation of the companies. For the inkjet codes, there could be various kinds of defects, such as the repetition, mistake, missing and blur due to the burn-in and malfunction of the inkjet printer or other external influences like the mechanical vibration. In the past, the quality of the code characters was inspected manually for randomly selected products. However, the limitations of the manual methods are self-evident, and they include high cost, and low efficiency, to name a few. Therefore, the defect detection of the code characters on the product package is of vital importance for the industry automation since it can increase the product quality, benefit the companies, and further protect the health of consumers.

Currently, optical and machine vision technologies are intensively used in the inkjet inspection machine to detect and recognize the code characters for the product quality check. These methods feature fast speeds, low cost and high accuracy. The commonly used defect detection methods in inkjet inspection machines are generally traditional machine vision methods. For these methods, the product's code image is firstly collected by the industrial camera, and single characters are located and segmented by image processing algorithms, such as binarization. Afterward, the segmented characters are sent to the recognizer for recognition. Then, the quality of the code characters is evaluated by the comparison between the recognized characters and the real characters. Therefore, the defect detection of the code characters can be summarized as a category of text detection [1]. To date, it becomes a very popular research field, and mature text detection methods have been developed to get good detection results. Typically, a feature extraction model composed of Gabor filter and Sobel operator to extract the character region and the K-means algorithm was used to distinguish the character regions from the background regions [2]. Then, a two-stage character segmentation framework has been developed with an accuracy of 99.92% in the vehicle license plate recognition, in which the AdaBoost algorithm was used to train a cascade classifier to locate key character regions and the position information of the key characters was utilized to predict the remaining characters [3]. Putro et al. proposed a real-time text detection method, in which the Sobel operator was used to perform edge detection on the image and the K-means was used to extract text from the image background [4]. To find text strings from arbitrary natural scene images, two methods have been proposed based on morphological operations with structure-based partitioning and morphology-based grouping [5].

Due to the rapid development of artificial intelligence recently, various solutions have been provided for the industrial automation. Deep learning can extract effective deep features from images by virtue of the convolutional neural networks, which has already shown great potentiality in tasks such as text detection. Tan et al. [6] proposed a character recognition method based on the corner detection and the convolutional neural network, which were implemented to mark candidate regions of text and recognize the separated text, respectively. Based on HCCR-GoogleNet, a simple and efficient character recognition network has been designed to construct a complete network architecture for oblique text correction, horizontal text line detection, character segmentation and character recognition [7]. Also, an end-to-end deep neural network has been proposed, in which the local information, global structural features and contextual prompt information for region-oriented suggestions were utilized to identify text instances [8]. Almost at the same time, another text detection method was developed to implement a custom faster RCNN and ResNet 50 convolutional neural network to locate the 320 × 240 text images and utilize a custom regression residual neural network to predict the text direction [9]. A new framework, which can run on mobile or embedded systems, realized the text line recognition based on two independent artificial neural networks and dynamic programming [10]. Cao et al. proposed a text detection method based on CTPN and presented an improved YOLO v3 network [11,12], the accuracy increased by 9.8% by changing the regression object from a single character to a fixed-width text and applying a stitching strategy to construct text lines based on the relation matrix. There is also a text structure feature extraction method for Chinese language based on the text structure component detector (TSCD) layer and residual network [13].

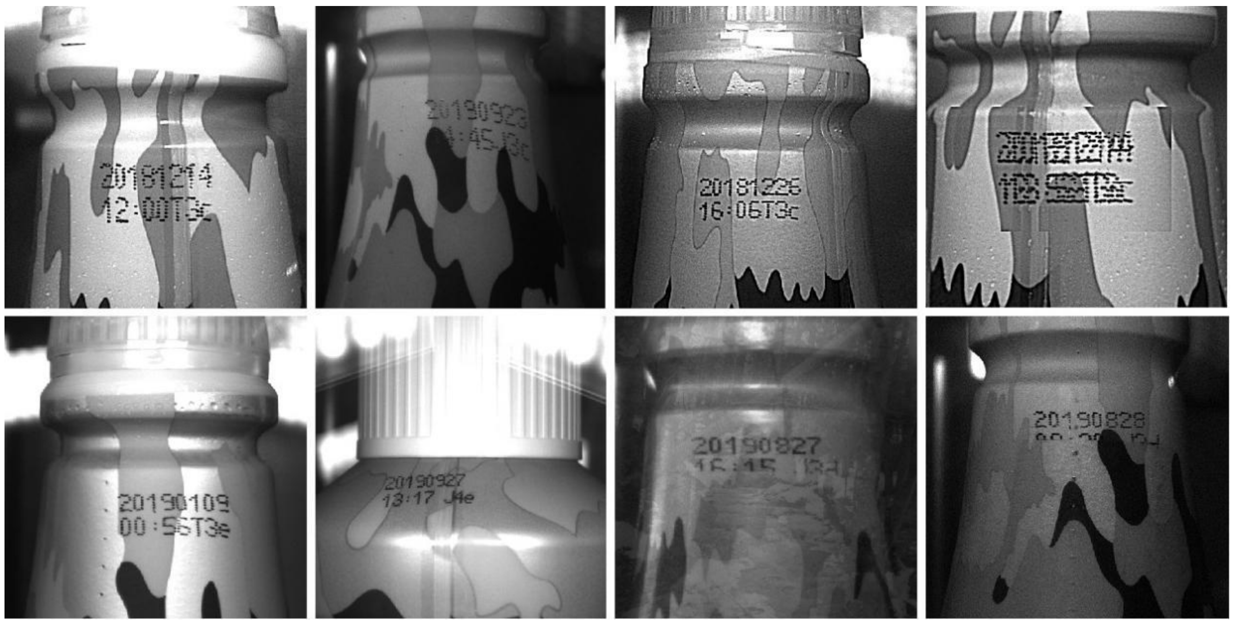


Figure 1. The code images with complex backgrounds on beverage package.

The objective of this paper is to develop an accurate and efficient detection system for the code characters by integrating the deep learning framework and the advanced object detection algorithm. The research subjects are the code characters on the beverage product package since the product lines in this field have become increasingly large with the development of economy and there is a strong need for the automatic quality inspection. In this paper, the code character pictures were directly from the industrial production line, in which the inkjet printer was used to print the codes of production date and the industrial camera was subsequently used to capture the area of codes. As shown in Figure 1, the backgrounds of the collected code images are so complicated that it is very challenging for traditional detection methods to perform accurate character segmentation. To clearly demonstrate the key features of the proposed defect detection system and the difference from the traditional detection method, its flow chart has been depicted in Figure 2. The overall detection system can be divided into three parts, namely the preprocessing of the code images, the design of the code defect detection algorithm and the design of the code defect detection software, as illustrated in Figure 2. In the experiments, a large number of image processing techniques were used to process the collected code images and construct the data set required. The convolutional neural network constructed the detection algorithm to extract the deep features of the code characters on complex backgrounds, and the detection model based on BBE has been trained with Python and Tensorflow to obtain the detector. Also, a friendly detect software has been developed based Python and PyQt5 to visually display the detection results and facilitate the operations for engineering personnel.

The rest of the paper is mainly organized as follows. Section 2 introduces the detection system platform and data preprocessing. In Section 3, the main network structure of the detection algorithm and the design of the detection process have been demonstrated with details. After that, there are some experimental data analysis and visual display of the results in Sections 4 and 5. Finally, the main conclusions and directions for further improvement have been summarized in Section 6.

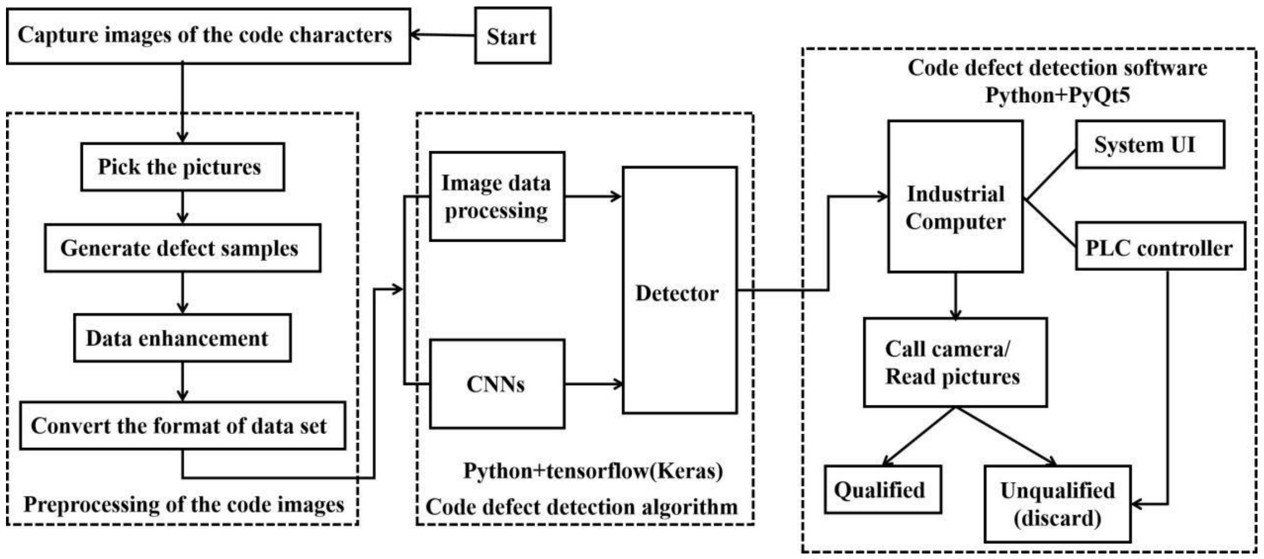


Figure 2. Flow chart of the overall code detection system.

2. System overview and image preprocessing

2.1. System overview

The code defect detection platform is generally set up behind the inkjet printer on the actual production line and the one used in this paper is shown in Figure 3, which mainly consists of an image acquisition unit, an industrial computer, a PLC controller, a power supply module and a human-computer interaction module. Specifically, the industrial computer eBOX-3622 is based on the Intel Core i5 multi-core processor, with powerful computing and image processing capabilities. The core component of the image acquisition unit is the Baumer VLG-02C.I industrial camera with high sensitivity and large dynamic range, which can maintain high image quality with a resolution of 656 × 490 at high frame rates. The power module is the energy core of the entire platform, and the PLC controller can control the mechanical devices. There is a human-computer interaction module with a touch screen featuring easy operation, which does not require mouse and keyboard.

For the image acquisition unit, it has been schematically illustrated in Figure 4. In this unit, the industrial camera was installed at the same height as the code characters on the beverage package, and the two bar-shaped light sources were installed at a 45-degree angle to the connection between the industrial camera and the beverage which for covering the code character area on the package. When the drink was delivered to the image acquisition unit, the photoelectric sensor triggered the light source to light up and the camera to take pictures, so as to collect the image of the code characters. Through these procedures, 1000 clean and clear code images have been obtained as the original data set.

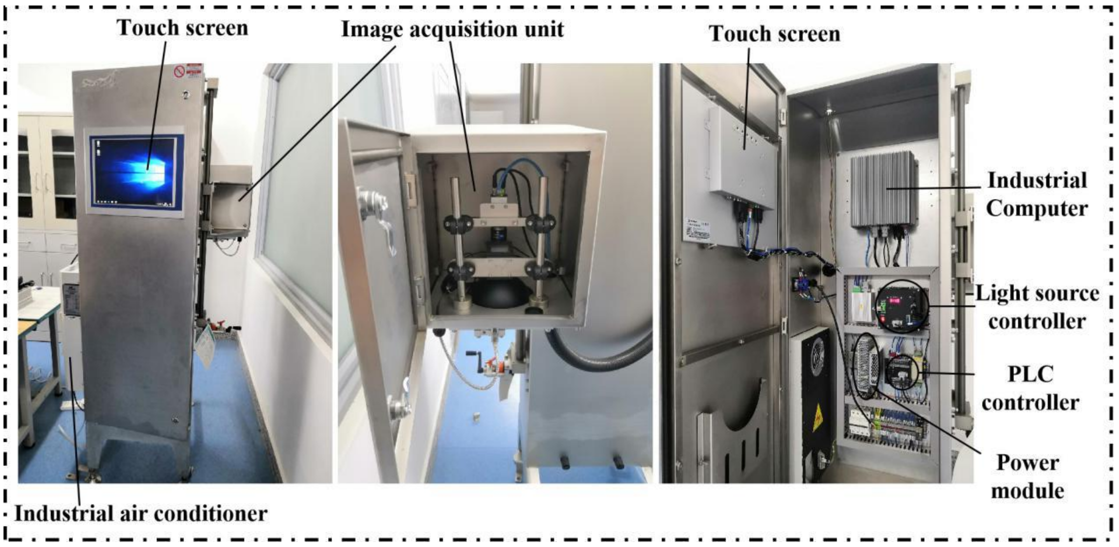


Figure 3. The platform of the code defect detection system.

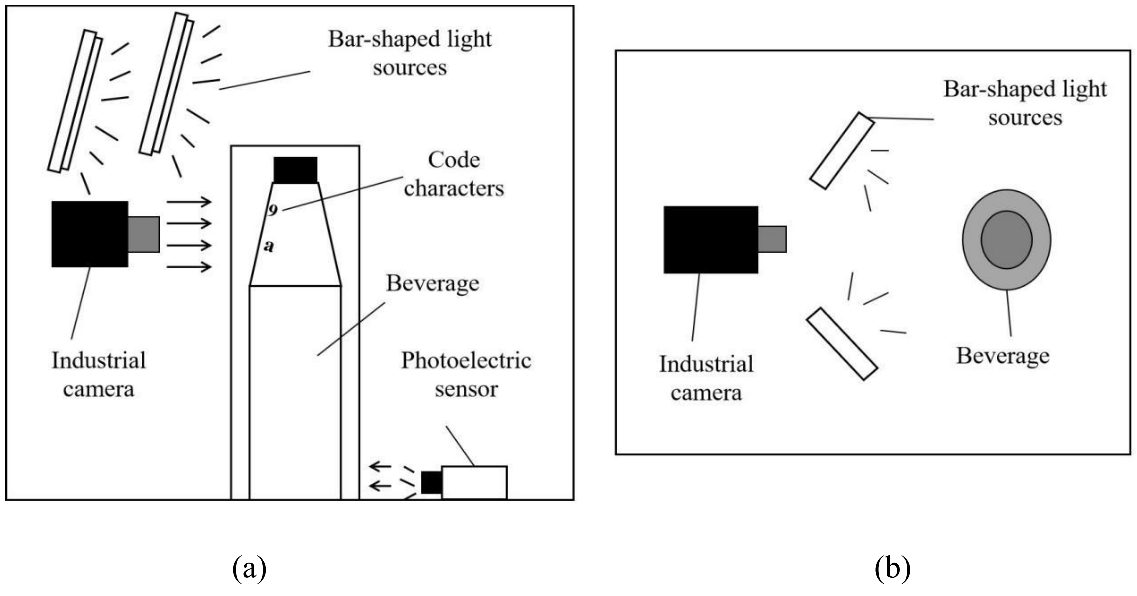


Figure 4. The structure diagram of image acquisition unit, (a) the front view, (b) the top view.

2.2. Image preprocessing

The shortcoming of the original data set is that the positive and negative samples are very uneven, and it is a small data set for deep learning. The image preprocessing in this paper mainly includes generating a large number of defective samples to balance the positive and negative samples, and further expanding the data set by virtue of data enhancement technologies.

2.2.1. Generating defect samples

The research object in this paper is the code characters of production date, which are two lines of black dot matrix characters on the beverage package. As a typical example, Figure 5 shows the defects for the beverage package, such as all code loss, partial code loss and code duplication in the actual production process.

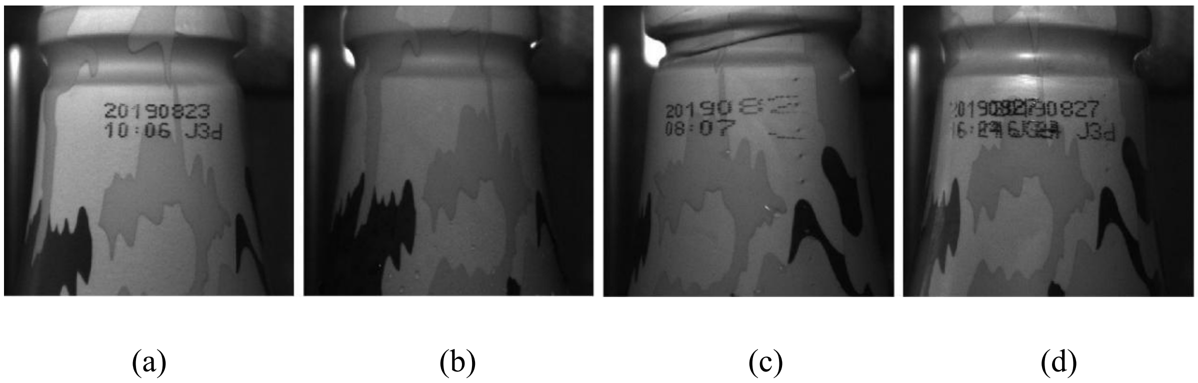


Figure 5. Some defect samples in the actual production process, (a) the case with correct code, (b), (c) and (d) the cases with all code loss, partial code loss and code duplication, respectively.

In this paper, the morphological operations have been used to process normal code images to generate defect samples and balance positive and negative samples [14]. The generated defect code images mainly include three types, code loss, code illegibility and code bonding, which were used to simulate and replace actual defect samples. For the morphological processing, its basic operations mainly include expansion, erosion, open operation and close operation. Assuming that there is an image *a* to be processed with a structural element *b*. The center point of the structural element is defined as the anchor point. The structural element was used to slide on the image and the anchor point pixel was replaced with the maximum value of the pixels covered by the structural element, which is the principle of the expansion. The expansion operation takes the maximum value in the neighborhood of each location, and therefore, the overall brightness of the expanded image will be higher than that of the original image. The brighter areas in the image will be further expanded and the darker areas will be further reduced or disappear. Contrary to the expansion operation, the point pixel will be replaced by the minimum value of the pixels covered by the structural element in the corrosion operation. In what follows, the bright areas will be further reduced or disappear, and the darker areas will be further expanded. The formulas for erosion and expansion operations can be expressed as

$$a \Theta b = \{(x,y) \mid (b)_{(x,y)} \subseteq a\} \tag{1}$$

$$a \oplus b = \{(x,y) \mid (b)_{(x,y)} \cap a \neq \phi\} \tag{2}$$

where *a* is the image to be processed, *b* is the structural element, (*x, y*) is the pixel to be processed, and ϕ is the empty set. The formulas for open operation and close operation are defined as

$$a \circ b = (a \Theta b) \oplus b \tag{3}$$

$$a \bullet b = (a \oplus b) \Theta b \tag{4}$$

respectively. It can be observed from Eqs (3) and (4) that the open operation means erosion first and then expansion, and the close operation indicates expansion first and then erosion. The expansion and open operations were used to process the normal code images to generate several defect samples, such as code loss, code illegibility and code bonding, as shown in Figure 6. The area where the code characters were located was selected for morphological processing of expansion and open operations. The expansion operation with the rectangular structure element was used to process the normal code image. If the size of the structure element is relatively large, most or all of the code characters would be removed to generate defective samples of code illegibility or all code loss. If the size of the element is relatively small, part of the code characters would be removed to generate defective samples of partial code loss. The open operation with the rectangular structure element was implemented to process the normal code images to generate defective samples of code bonding.

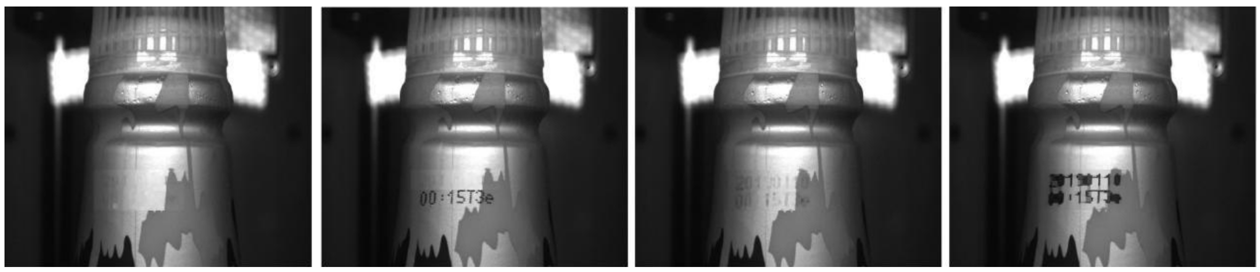
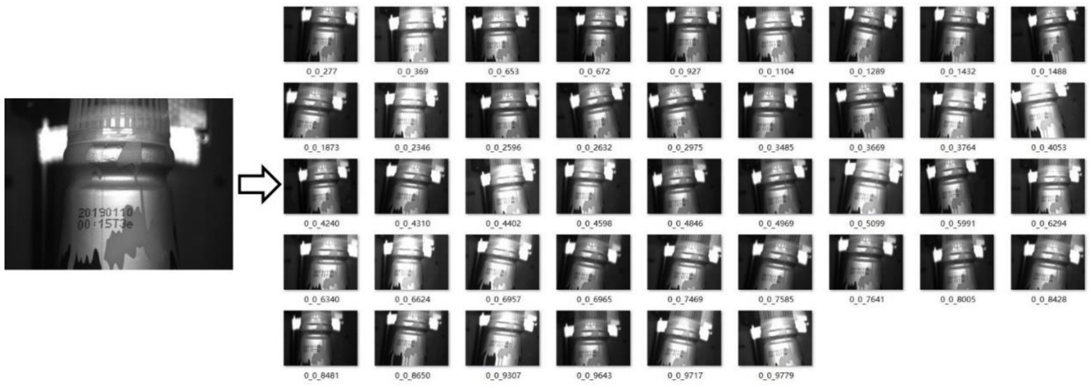

(a) (b) (c) (d)

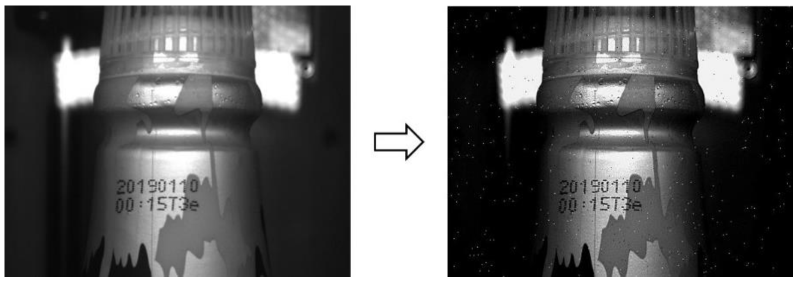
Figure 6. Some defect samples generated by morphological operations, (a) the case with all code loss, (b), (c) and (d) the cases with partial code loss, code illegibility and code bonding, respectively.

2.2.2. Image enhancement

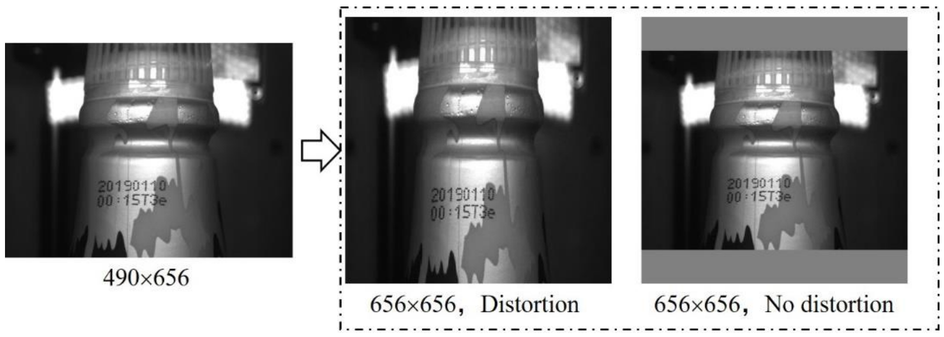
Contrary to that of the image classification task, the training data of the detection task includes the rectangular box labels of the manually labeled target objects. When performing image enhancement, it is not sufficient to only enhance the image directly, and it also needs to consider the change of the rectangular box labels after the image is changed. These rectangular box labels have the same enhancement effect as the image, which mainly includes the enhancement of translation, cropping, zooming, inversion, changing brightness and contrast, etc. Figure 7(a) shows the above enhancement effects, and Figure 7(b) is the enhancement effect of adding noise to the original image. As shown in Figure 7(c), The integrity of the image is maintained by adding gray bars to avoid image distortion during processing due to the fact that original image is not in a square shape and the input image of the neural network generally requires the same length and width. It should be mentioned that the resolution of the original code image is 656 × 490. In this paper, three sizes of the code images with resolutions of 320 × 320, 384 × 384 and 448 × 448 were designed for training. As illustrated in Figure 8(a), the white rectangular boxes display the corresponding change as the image itself is enhanced and the actual image enhancement will process the rectangular boxes of all characters.



(a)



(b)



(c)

Figure 7. Processes before image enhancement, (a) some enhancement effects, (b) enhancement effect of adding noise to the original image, (c) adding gray bars to avoid image distortion during processing.

To further strengthen the image enhancement effect, multiple enhanced code images were combined together. Specifically, the random image enhancements in Figure 8(a) have been performed on multiple images. The images and their label boxes were placed and combined in multiple directions to get the combined image, which has the same size with the original code image. The advantage of this combination is that it can enrich the background of the detection target and enhance the detection effect on the target object, which is small or on the complex background. Moreover, deep neural networks can read multiple images simultaneously during training and calculation, which speeds up the training and calculation processes. Figure 8(b) shows four code images, which were combined for image enhancement. The white rectangular boxes visually display the changes made by all characters in each image, and the combined code image has richer background.

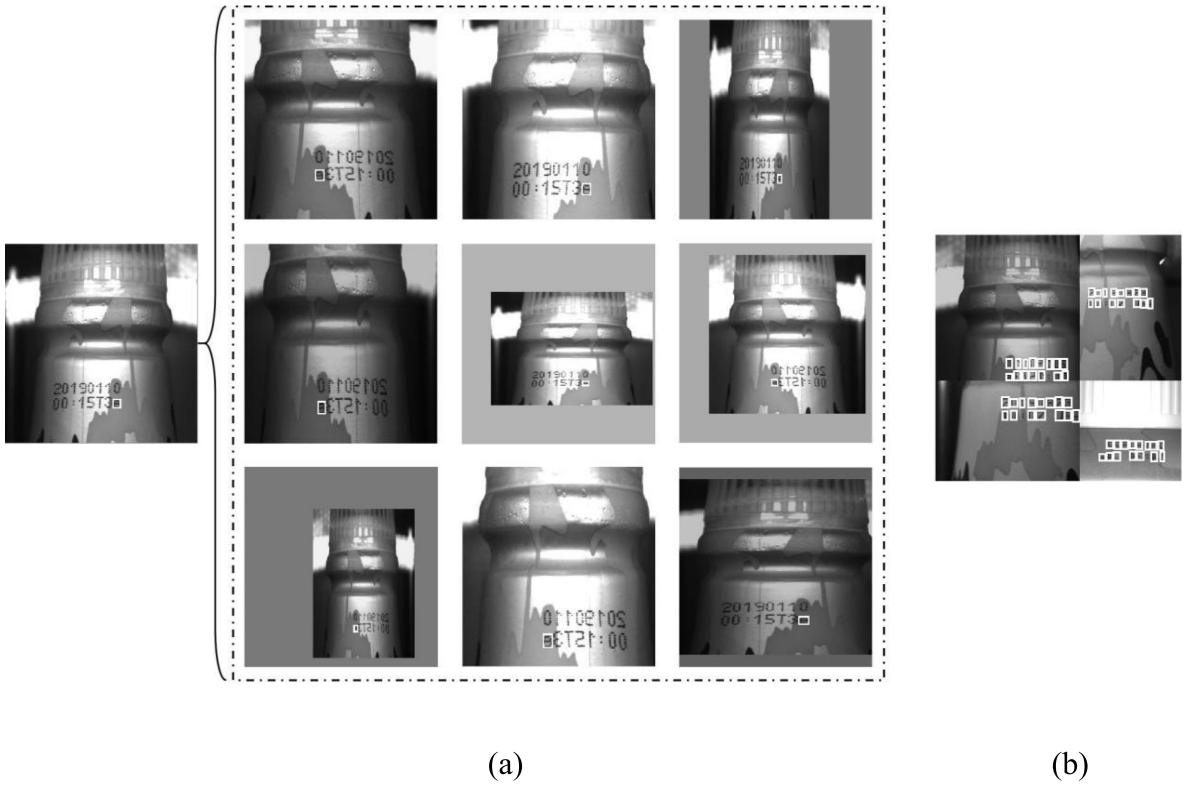
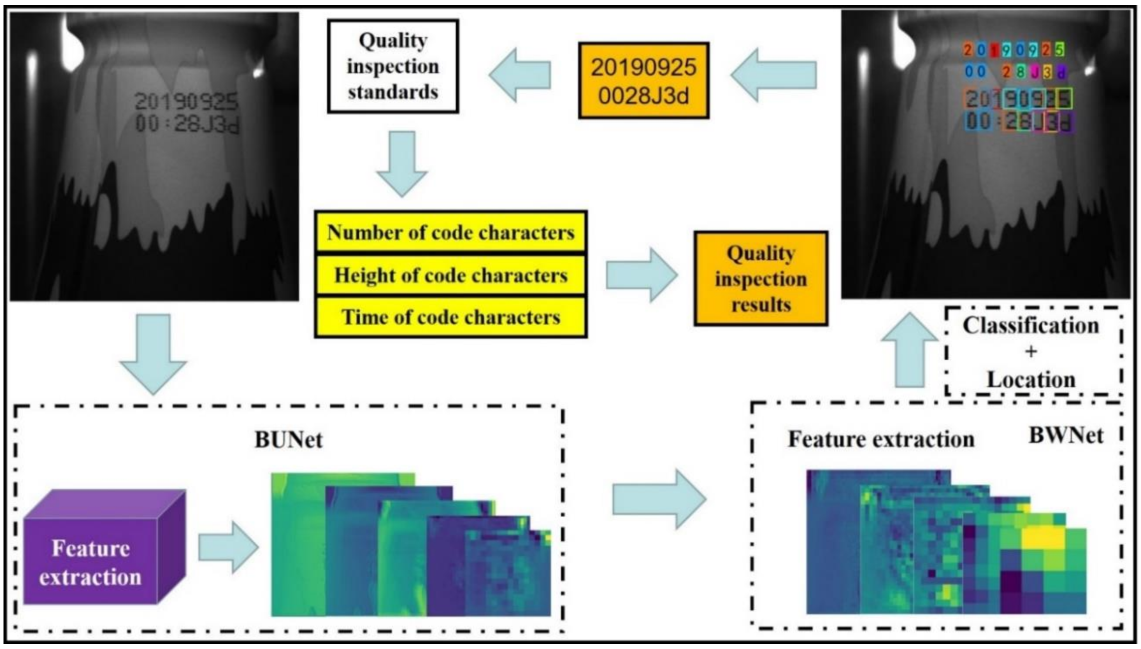


Figure 8. Examples of image enhancement, (a) random enhancements in the code image and its box labels with the white rectangular boxes displaying the random changes, (b) random combination of four enhanced images.

3. Methodology

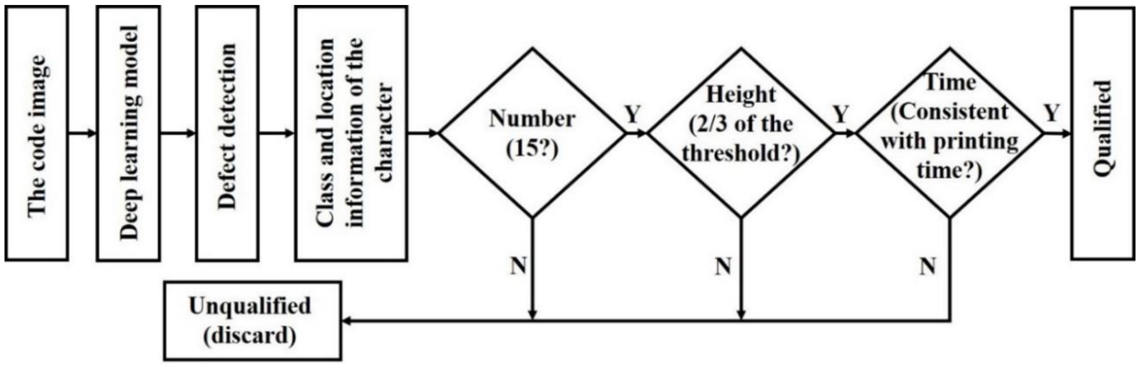
3.1. The overall structure

The proposed code character detection system has been designed based on object detection [15] with its overall architecture diagram shown in Figure 9. The overall detection process can be mainly divided into two steps. Firstly, the designed object detection algorithm was used to classify and locate the characters on the code image. Then, the relevant quality inspection standards were set to filter the detection results of individual characters and evaluate the overall code image quality. The object detection algorithm is mainly divided into a feature extraction network called BUNet, a feature fusion network called BWNet, a classification and location network. BUNet is used to extract the depth features of the coded characters, BWNet is used to refine and filter the extracted features, classification and location networks are used to obtain the class and location information of the code characters. The construction of BUNet draws on the core module of EfficientNet [32], that is, the object detection network in this paper is mainly based on EfficientNet, BUNet and BWNet, which is named BBE. More specifically, the original data set was preprocessed to obtain the training data set and the object detection model was trained and tested. Afterward, the detection model was implemented to detect the input image to obtain the classification and location results, i.e., the class and position information of the code characters. If the number of code characters is the preset value (15), the height of the code character reaches 2/3 of the standard threshold and the time represented by the code characters are the same as the printing time, which means that the code characters are qualified. Otherwise, it indicates that the code characters are unqualified and need to be discarded. The flow chart of the steps of code character defect detection is as Figure 10.



Quality inspection standards
20190925 0028J3d
Number of code characters
Height of code characters
Time of code characters
Quality inspection results
Classification + Location
BUNet
Feature extraction
Feature extraction
BWNet

Figure 9. The overall architecture diagram of the detection method.



The code image
Deep learning model
Defect detection
Class and location information of the character
Number (15?)
Height (2/3 of the threshold?)
Time (Consistent with printing time?)
Y
Y
Y
Qualified
N
N
N
Unqualified (discard)

Figure 10. Flow chart of the steps of code character defect detection.

3.2. The object detection network called BBE

In recent years, the development of deep learning has secured its leading position in the field of image processing, which largely depends on the powerful feature extraction capabilities of convolutional neural networks. Evolving from the first convolutional neural network LeNet-5 [16], AlexNet [17], VGGNet [18], GoogleNet [19] to ResNet [20], SqueezeNet [21], Xception [22] and MobileNet [23] series, deep neural networks now feature smaller calculations, higher accuracies and faster speeds, which constantly refresh the applications of image processing in the industrial field and provide a large number of novel solutions. For the structure of the deep convolutional neural networks, it is mainly composed of convolutional layers, pooling layers and fully connected layers. The more convolutional layers, the stronger the network's ability to extract deep features. Similar to the BP neural network [24], the convolutional neural networks utilize the forward propagation to calculate the output value and implement the backward propagation to adjust the weight and bias. The difference is that the convolutional neural network has a feature extractor composed of a convolutional layer and a pooling layer, which extracts features and down-samples these features into more abstract features, respectively. It should be stressed that the main features of convolutional neural networks are weight sharing and local receptive fields. The implementation of the weight sharing and the local receptive field can greatly reduce the number of parameters that the convolutional neural network needs to train, so that overfitting is not easy to occur during the training process with the generalization ability of the model improved.

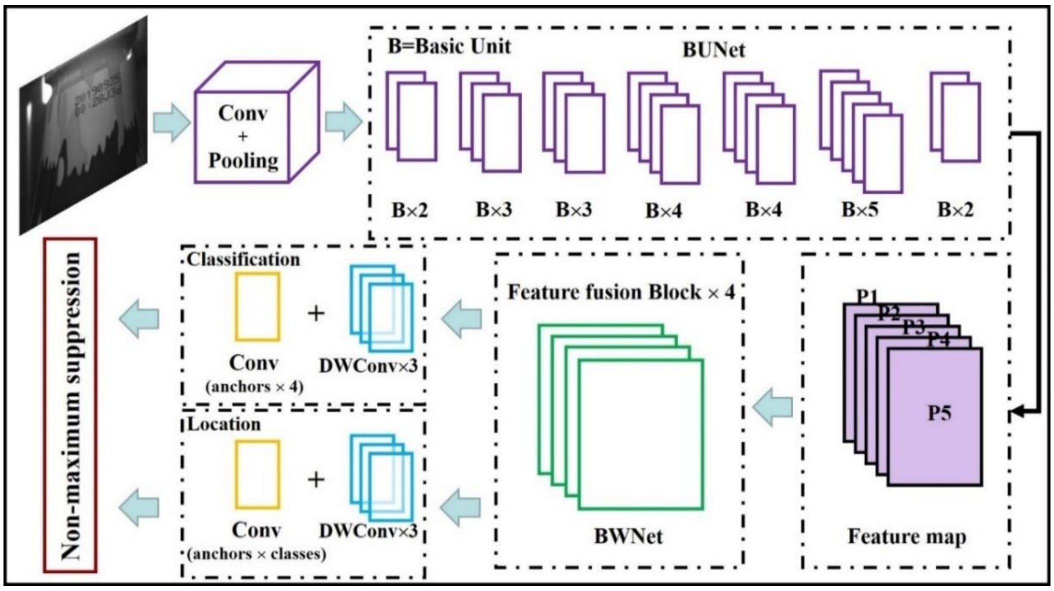


Figure 11. The structure frame diagram of BBE.

The object detection algorithm based on deep learning mainly includes the two-stage method and the one-stage method, both of which need to complete the tasks of classification and location. The typical two-stage methods are RCNN [25], SPPNet [26], Fast-RCNN [27], Faster-RNN [28] and Mask-RCNN [29]. In these methods, the convolutional networks are used to process the image to generate a large number of candidate regions, which are sent to a higher-level network for processing to get more refined information of classification and location. For the one-stage method, it mainly includes the series of YOLO [30] and SSD [31]. Its key idea is to perform dense sampling on the feature map of the image and generate a large number of prior boxes that can be directly classified and located. Overall, the two-stage method has high accuracy and relatively slow speed. On the contrary, the one-stage method has fast speed and relatively low accuracy due to that it discards the structure of generating candidate regions.

In this paper, the BBE object detection network with high accuracy and speed is constructed based on the core module of EfficientNet [32], and the details of which have been schematically shown in Figure 11. As shown in Figure 11, the backbone BUNet consists of 7 basic blocks (23 basic units in total) stacked to extract deep features of code characters on complex backgrounds. Then feature fusion was performed on the obtained effective feature maps. The feature fusion network BWNet is composed of 4 feature fusion blocks, which are used to filter and refine the 5 effective feature maps obtained from BUNet. The classification and location results were obtained with the detection of classification and location network. The three 3 × 3 depthwise separable convolution layers used to organize the five effective feature maps from BWNet are the same structure of the classification and the location network. The last layer of the classification and location network is the depthwise separable convolution with the number of channels of 9 × 16 and 9 × 4, which are used to obtain the class and location information of the code characters, respectively. The number of anchors, classes of code characters and adjustment parameters of each anchor are 9, 16 and 4, respectively. The final detection result is obtained by performing score sorting and non-maximum suppression screening on the detection results of classification and location. The BBE network can be briefly summarized as Figure 12(a), where BUNet and BWNet constitute its main structure.

The depthwise separable convolution, which consists of the depthwise convolution and the pointwise convolution, has been widely used to extract features in some lightweight networks [22]. The calculation principle of the depthwise separable convolution is shown in Figure 12(b). In this structure, the 1 × 1 pointwise convolution was used to increase or reduce the dimension of the feature channel, and the 3 × 3 depthwise convolution was utilized to extract the depth features.

The basic unit in BUNet is roughly based on the inverted bottleneck structure shown in Figure 12(c), which is similar to the core module called Efficient Block in EfficientNet. This structure has stronger feature extraction capability than the classic residual structure [20], and some optimization strategies are added to it to further improve the performance of the network. The optimization strategies mentioned are as follows: 1) The depthwise separable convolution is improved. Specifically, the number of channels is adjusted first, and then the features are extracted. The 3 × 3 depthwise convolution is split into two asymmetric convolutions of 1 × 3 and 3 × 1; 2) An attention mechanism is applied to the feature channel after the feature extraction is completed, which allows the network to learn and pay more attention to the channel where the effective feature is located. 3) Linear convolution is used directly instead of the Swish activation function to retain more feature information after the 1 × 1 convolution is used to reduce the number of feature channels.

In this paper, the proposed network chose the Swish [33] function as the activation function, which states that

$$f(x) = x \cdot sigmoid(\beta x) \tag{5}$$

with the sigmoid function widely used in logistic regression and neural networks [34] given by

$$f(x) = \frac{1}{1+e^{-x}} \tag{6}$$

It can be observed that the Swish function can be regarded as a smooth function between the ReLU function [35] and the linear function with the adjustment of the hyperparameter β.

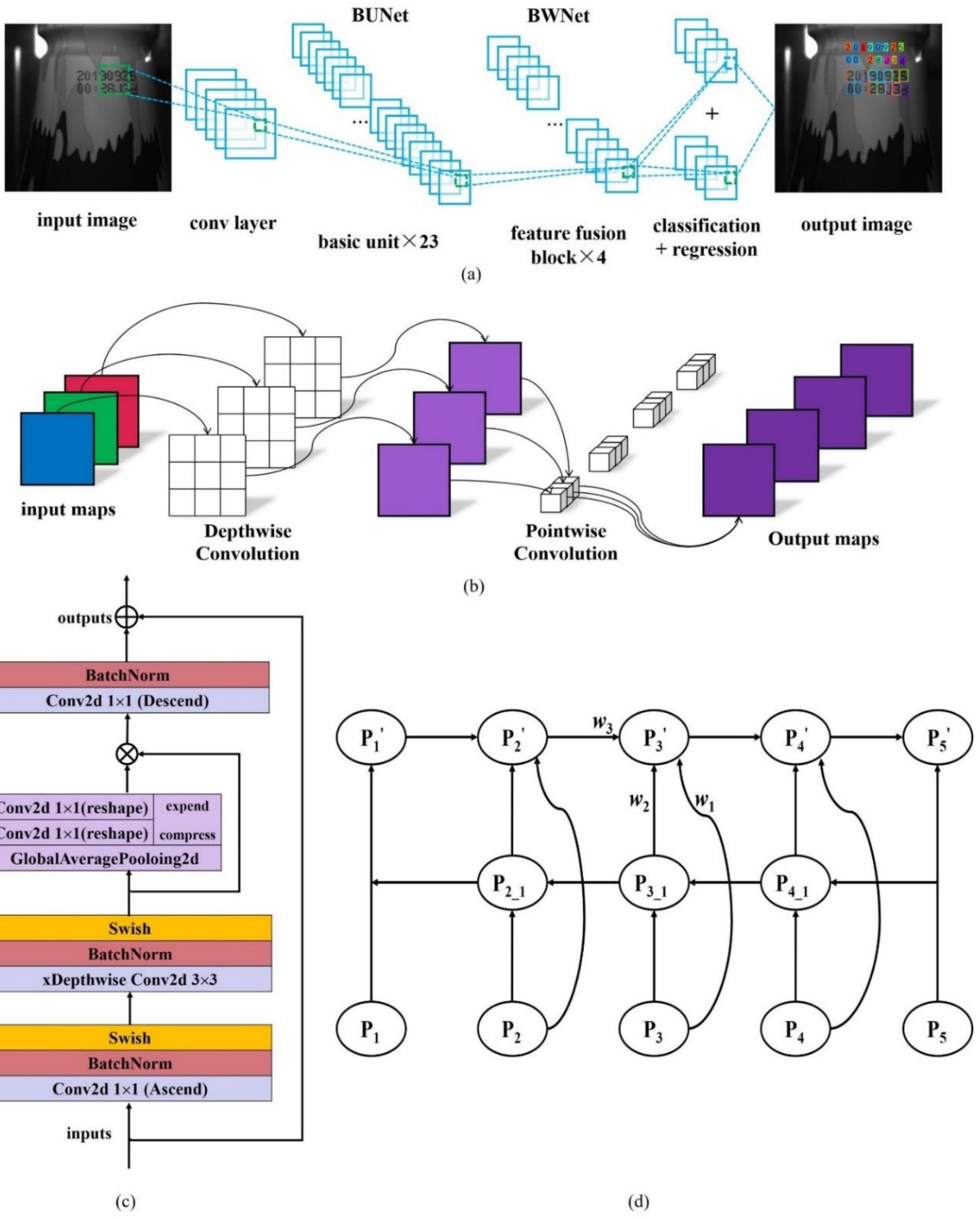


Figure 12. The schematic diagram of the detection network, (a) its structure, (b) the calculation diagram of the depthwise separable convolution, (c) the structure of the basic unit, and (d) the structure of the feature fusion block.

In the proposed network, the Batch Normalization (BN) layer has been used to optimize the distribution of input data in each layer to reduce the training time of the network and accelerate the convergence of the model [36]. The mathematical expression of BN is given by

$$y = \gamma \frac{x-\mu}{\sqrt{\sigma^2+\varepsilon}} + \beta \tag{7}$$

where μ and σ are the mean and variance of the small batch training data, γ and β are reconstruction parameters can be learned, and ε is a small constant.

The feature fusion network in this paper is composed of 4 feature fusion blocks, as shown in Figure 12(d), where P_1~P_5 are the five efficient feature maps from the feature extraction network, P_{2_1}~P_{4_1} are the feature maps of the middle layer, and P_1'~P_5' are the output feature maps. The feature fusion block continuously performs up-down sampling and weighted fusion on the extracted features, which can effectively alleviate the problem of feature information loss caused by the shallow feature information transmitted to the very deep layer. Similar to the structure used in literature [37], the deep effective feature maps very close to the code characters could be obtained. Taking P3 as an example, its weighted fusion method is expressed as

$$P_3' = \frac{P_3 \times w_1 + P_{3\_1} \times w_2 + re(P_2') \times w_3}{w_1 + w_2 + w_3 + c} \tag{8}$$

where w_1, w_2 and w_3 are the fusion weights of different feature nodes learned and trained with neural networks, and c is a small value to ensure the stability of the numerical result. It should be mentioned that feature scaling is required before fusion due to the different sizes of feature maps at different levels.

4. Discussion

4.1. Environmental configurations

For the experiments conducted in this section, the configuration details are mainly listed as follows. Ubuntu served as the operating system with the Tensorflow 2.2.0 [38] and its encapsulated Keras 2.3.1 as the deep learning framework. Python 3.7 and Spyder were used as the programming language and the integrated development environment (IDE), respectively. The CUDA 10.1 and CUDNN 7.6.5.32 were utilized to accelerate the training of the model, and a single GTX 1080TI graphics card was used as the acceleration hardware.

The callback function ReduceLROnPlateau in Keras was used to optimize the learning rate, and the attenuation of the learning rate was adjusted according to

$$lr = lr' \times factor \tag{9}$$

where lr and lr' are the learning rates before and after the update, respectively, with factor being the scaling factor. The initial learning rate and the scaling factor were set to 0.001 and 0.75 respectively, and the training batch size was 16. The learning rate would reduce once if the model was trained for two epochs and the performance was not improved. The model would stop training if the model was trained for six epochs and the performance was not improved.

In the proposed network, the Adam algorithm [39] worked as the optimizer to update various training parameters, which is an extension of the stochastic gradient descent (SGD) [40] method with adaptive learning rate to accelerate the convergence speed of the model. Also, the hyperparameters only need a very small amount of adjustment to solve most of the training problems in the update process of weights and biases.

4.2. Experimental results

In this paper, the proposed network implemented several evaluation indicators for performance evaluation, including precision, recall, F1 score and accuracy, the definitions of which can expressed as, respectively,

$$Precision = \frac{TP}{TP + FP} \tag{10}$$

$$Recall = \frac{TP}{TP + FN} \tag{11}$$

$$F_{1\_}score = 2 \cdot \frac{Precision \times Recall}{Precision + Recall} \tag{12}$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{13}$$

Here, *Precision* is the probability of actually being a positive sample among all samples that are predicted to be positive. *Recall* is the probability of being predicted as a positive sample in the actual positive samples. *F1_score* is the weighted average of precision and recall. *Accuracy* is defined as the proportion of correctly classified samples to the total samples. *TP* and *TN* represent true positive cases and true negative cases, and *FP* and *FN* represent false positive cases and false negative cases, respectively. Based on the above indicators, the mAP (mean Average Precision) refers to the average value of AP (average precision) of all detection categories, and AP is the area under the curve of PR (precision and recall) of this category. mAP is the most important indicator in object detection, which can accurately reflect the accuracy information of the model.

It should be mentioned that the proposed network is based on a modular design and its feature extraction capability can be modified according to the task needs with different object sizes. Table 1 shows that the impact of different basic units on BUNet, BUNet_v1~BUNet_v3 are composed of 16, 23 and 26 basic units respectively. The BUNet_v2 was used to as the final BUNet due to the fact that it not only guarantees high accuracy but also facilitates the detection of small targets such as code characters.

Table 1. The impact of different basic units on BUNet.

Indicator	BUNet_v1	BUNet_v2	BUNet_v3
Accuracy	0.9352	0.9452	0.9584
Parameters	2.7M	5.4M	10.8M

The feature fusion network BWNet performs further deep feature extraction and feature fusion on five different scale deep effective feature maps from the feature extraction network BUNet. It solves the problem that a large amount of feature information will be lost when the shallow feature information is transferred to the deep network. Table 2 shows the influence of the presence or absence of feature fusion network on the overall algorithm. The BWNet network brings a performance improvement of close to 5.3% to the mAP and accuracy, and the overall model is still an acceptable lightweight model.

Table 2. The impact of the feature fusion network on the algorithm.

Indicator	Remove BWNet	Complete model
mAP	0.9432	0.9961
Accuracy	0.9453	0.9985
Parameters	5.4M	6.6M
Storage space	25.9M	27.1M

Linear convolution means that in the design process of the basic unit, the depthwise separable convolution is improved, the attention mechanism is applied, and finally the linear convolution is used to replace the Swish activation function for the activation operation. Table 3 demonstrates the effect of using linear convolution or the Swish activation function in the basic unit for activation operation on the algorithm. Linear convolution can retain more feature information and further improve the mAP and accuracy of the model.

Table 3. The effect of linear convolution on the model.

Indicator	Use Swish activation function	Use linear convolution
mAP	0.9841	0.9961
Accuracy	0.9863	0.9985
Parameters	6.6M	6.6M

The Pascal VOC 2012 data set was used as the pre-training data set. Table 4 demonstrates the impact of different pre-training methods on network performance, where pre-training 1 used BUNet+BWNet for training and pre-training 2 used BUNet for training. Pre-training 1 was the training method in this paper.

Table 4. The impact of different pre-training methods.

Indicator	Pre-training 1	Pre-training 2	No pre-training
mAP	0.9961	0.9925	0.9823
Accuracy	0.9985	0.9937	0.9845
Storage space	27.1M	25.9M	27.1M

Table 5. Experimental results of the BBE network.

Method	BUNet	BWNet	Linear convolution	Pre-training	mAP	Parameters
EfficientNet	×	×	×	×	0.9389	5.3M
BBE	√	×	×	×	0.9431	5.4M
BBE	√	√	×	×	0.9713	6.6M
BBE	√	√	√	×	0.9833	6.6M
BBE	√	√	√	√	0.9961	6.6M

The experiments done in this section can be summarized in Table 5. Linear convolution refers to that in the design process of BUNet, the basic unit used linear convolution instead of the Swish function for activation and dimensionality reduction operations.

In the experiments, there were 3690 code images used in the data set, among which the ratio of the train set to the verification set was 9:1. Various performance indicators were tested on the test data set with 675 code images, which is different from the training data. The training details of the model are shown in Figure 13 and Table 6.

As shown in Figure 13 and Table 6, the convergence speed of the model was initially fast and then became slow during the training process. Afterward, the model converged to a saturated state after training for about 40 epochs, with the mAP reaching 0.9961.

Table 6. Various indicators of the model during training.

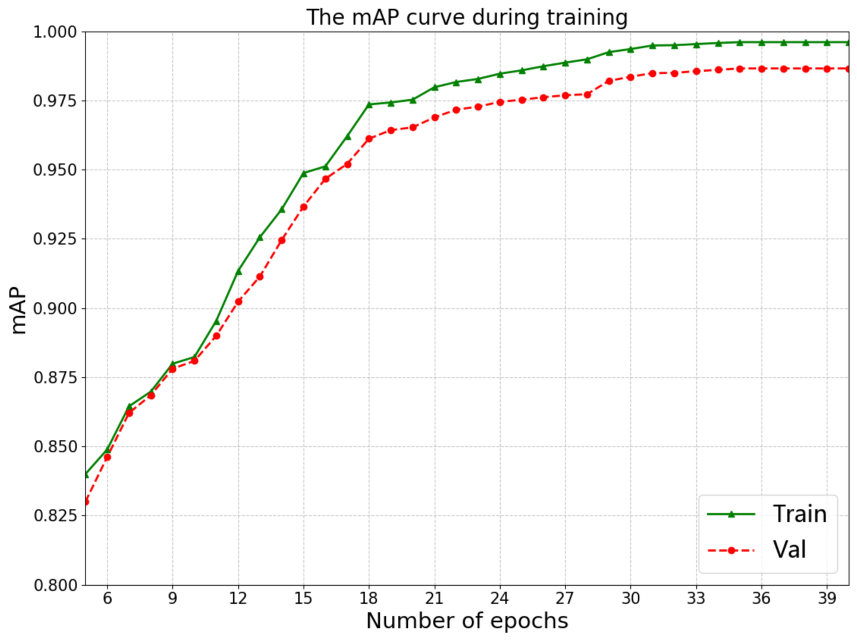
Epochs	Precision	Recall	F1-score	mAP
10	0.9588	0.6894	0.8021	0.8849
20	0.9820	0.8818	0.9292	0.9722
30	0.9896	0.9521	0.9705	0.9925
40	0.9812	0.9691	0.9751	0.9961

Table 7. Experimental results of different models.

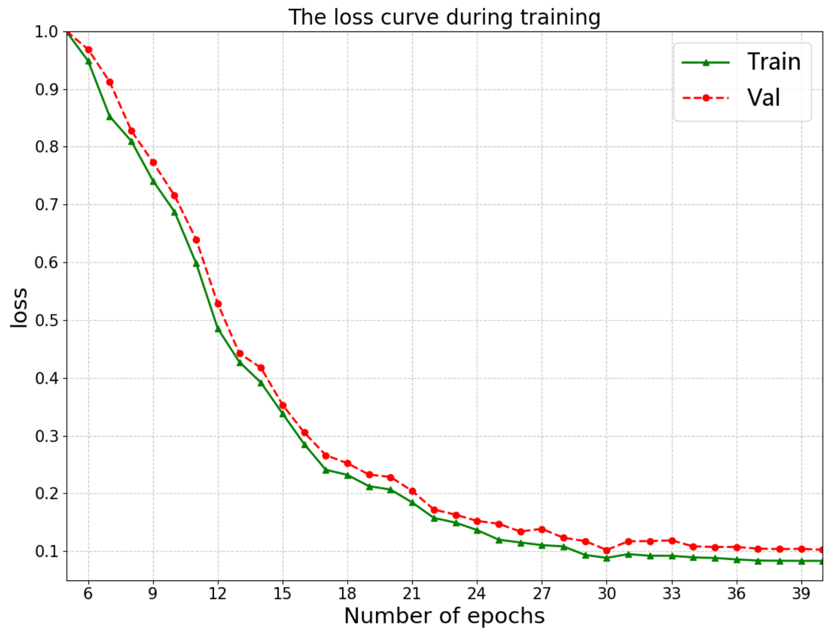
Method	mAP	Accuracy	Parameters	Speed	Storage space
Faster RCNN	0.9530	0.9624	28.5M	785 ms	109M
SSD	0.9527	0.9612	25.8M	283 ms	103.1M
YOLO V3	0.9631	0.9724	62.6M	94 ms	237M
MobileNet-SSD	0.9615	0.9702	8.5M	138 ms	34.1M
YOLO V4_tiny	0.9717	0.9823	6.1M	40 ms	23.1M
YOLO V5	0.9930	0.9950	7.3M	12 ms	14M
CenterNet	0.9656	0.9680	32.7M	61 ms	125M
Ours	0.9961	0.9985	6.6M	72 ms	27.1M

As shown in Table 7, the classic object detection methods and their related lightweight models were compared to the proposed method. These classic methods include Faster RCNN [28], SSD [31] and YOLO V3 [41] and the lightweight models include MobileNet-SSD [23], YOLO V4_tiny [42] and YOLO V5 [43]. CenterNet [44] is different from common object detection network, and the detection object is regarded as a point during the process of constructing the model. It should be mentioned that MobileNet-SSD, YOLO V4_tiny and YOLO V5 have further improved detection accuracy and speed on the basis of reducing the volume of the model. According to the experimental results shown in Table 7, the method proposed in this paper has the good comprehensive performance, specifically, the highest accuracy and a relatively high speed.

To intuitively demonstrate the superior performance of the proposed network, some existing code detection methods have been included in the experiments with the corresponding results listed in Table 8. These methods are mainly based on traditional visual detection or the combinations of traditional methods and neural networks, which is different from the pure deep convolutional neural networks for the end-to-end detection. As illustrated in Table 8, the detection accuracy of the existing methods is very high. Most of them reach more than 95% and some even reach 99%. Still, these accuracies are generally below that of the proposed network. The main limitation of the existing methods is the relatively small application scope, since most methods can only detect the code image with simple background or high contrast with generally slow detection speeds. Therefore, it is difficult for these methods to achieve a balance between accuracy and speed.



(a)



(b)

Figure 13. (a) The mAP curves during training, (b) loss curves during training respectively.

Overall, the proposed method in this paper is an end-to-end detection method with better accuracy and speed, and it has strong learning ability and adaptability. Furthermore, it is simple to maintain since it only needs to set the parameters and train the model according to the requirements without adjusting the parameters repeatedly. Therefore, the proposed method can greatly reduce the maintenance cost and improve the detection efficiency compared to the traditional detection method featuring manually parameters adjusting.

Table 8. The performance comparison of different code detection methods.

Method	Description	Research object	Date	Accuracy	Speed
Chen et al. [45]	Image mosaic & String match & CNNS	Inkjet code on tire mold (Simple background)	2017	0.9967	106.5ms
Yang et al. [46]	Bacteria foraging optimization algorithm & SVM	Printed character (High contrast)	2019	0.968	< 150 ms
Sun et al. [47]	Line scan & vertical projection segmentation & LeNet-5	Spray code in milk production (Complex background)	2019	0.9217	9.7 ms
Zhou et al. [48]	Template matching & gray projection	Bottom spray code of daily chemical bottles (Simple background)	2020	0.97	620 ms
Wang et al. [49]	SIFT features-based image matching & Faster RCNN	Character in piston cavity (Simple background)	2020	0.99	--
Ours	EfficientNet & weighted feature fusion & Defect detection	Code character on drink package (Complex background)	2021	0.9985	72 ms

4.3. Visualization

4.3.1. Deep features

For the proposed detection method in this paper, the convolutional layers mainly exist in BUNet and BWNet. As illustrated in Figure 14, P_1~P_5 are the five deep effective feature maps generated by the feature extraction network and the depth deepens in turn, and P_1'~P_5' are the output feature maps of the feature fusion network corresponding to P_1~P_5. It can be observed that the extracted features are more abstract and representative with the increase of the network layers, due to that the feature extraction ability of the convolutional neural network became stronger. The five deep effective feature maps generated by the feature extraction network should continue to be put into the feature fusion network for further deep feature extraction. The feature fusion network performs deep feature extraction by continuously weighting and fusing different feature maps to extract the smaller and more abstract feature maps, which are closer to the effective features of the code characters in the image.

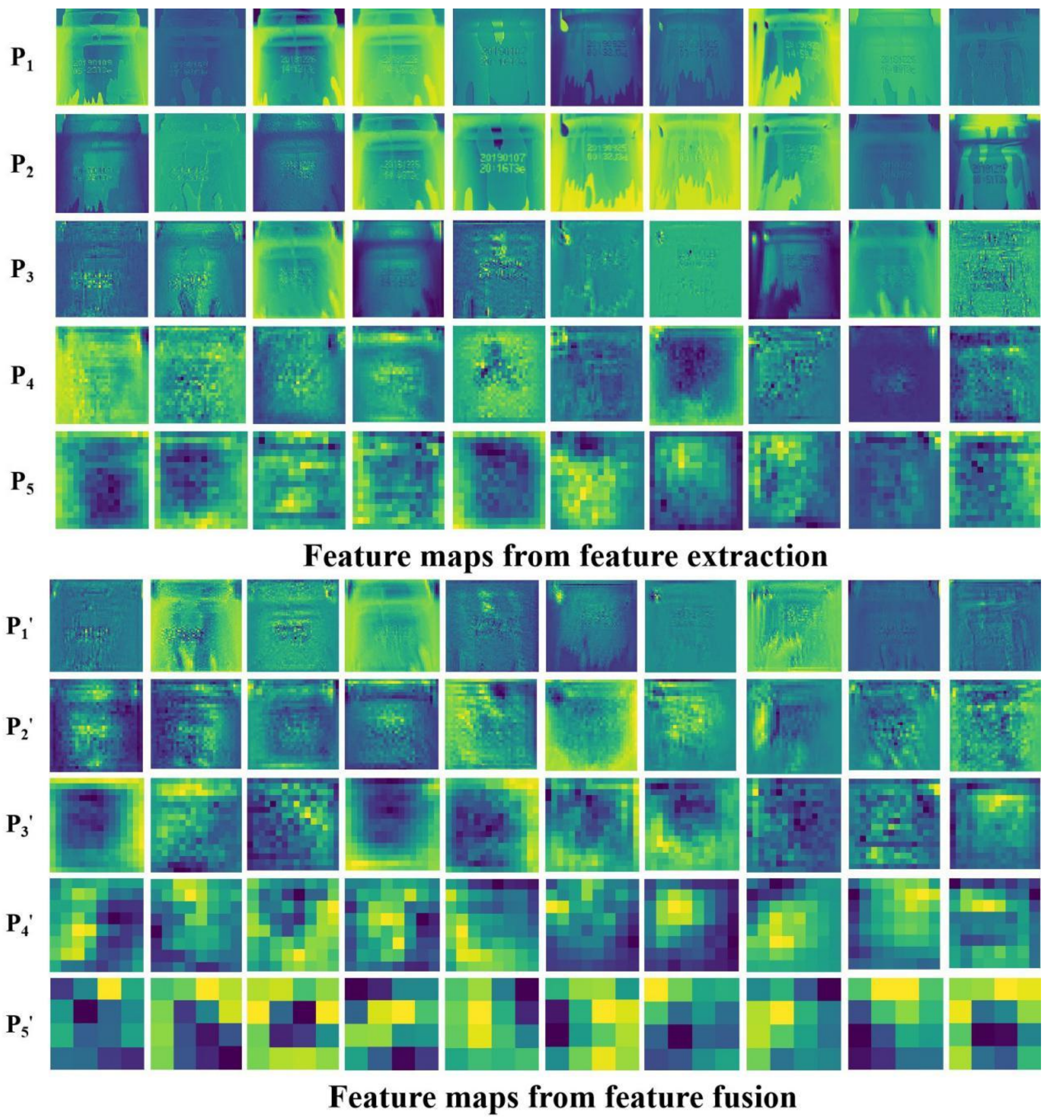

P_1
P_2
P_3
P_4
P_5
Feature maps from feature extraction
P_1'
P_2'
P_3'
P_4'
P_5'
Feature maps from feature fusion

Figure 14. The output feature maps from feature extraction and feature fusion, where P_1~P_5 become more and more abstract and P_1'~P_5' become closer to the effective features of the code characters in the image.

4.3.2. Detection results

The experimental results of the code defect detection have been shown in Figure 15, in which the first half is the detection results of qualified code images and the second half is the detection results of defective code images. As illustrated in Figure 15, the detection method proposed in this paper can accurately identify the classes of the code characters and locate the positions. Also, it demonstrates strong capabilities of feature extraction and processing, which can extract effective features of code characters on complex backgrounds and quickly obtain accurate detection results. In what follows, the quality of the product can be quickly and accurately evaluated by setting quality inspection standards for the classification and location results obtained from the inspection of the code image, which greatly improves the efficiency of industrial production.

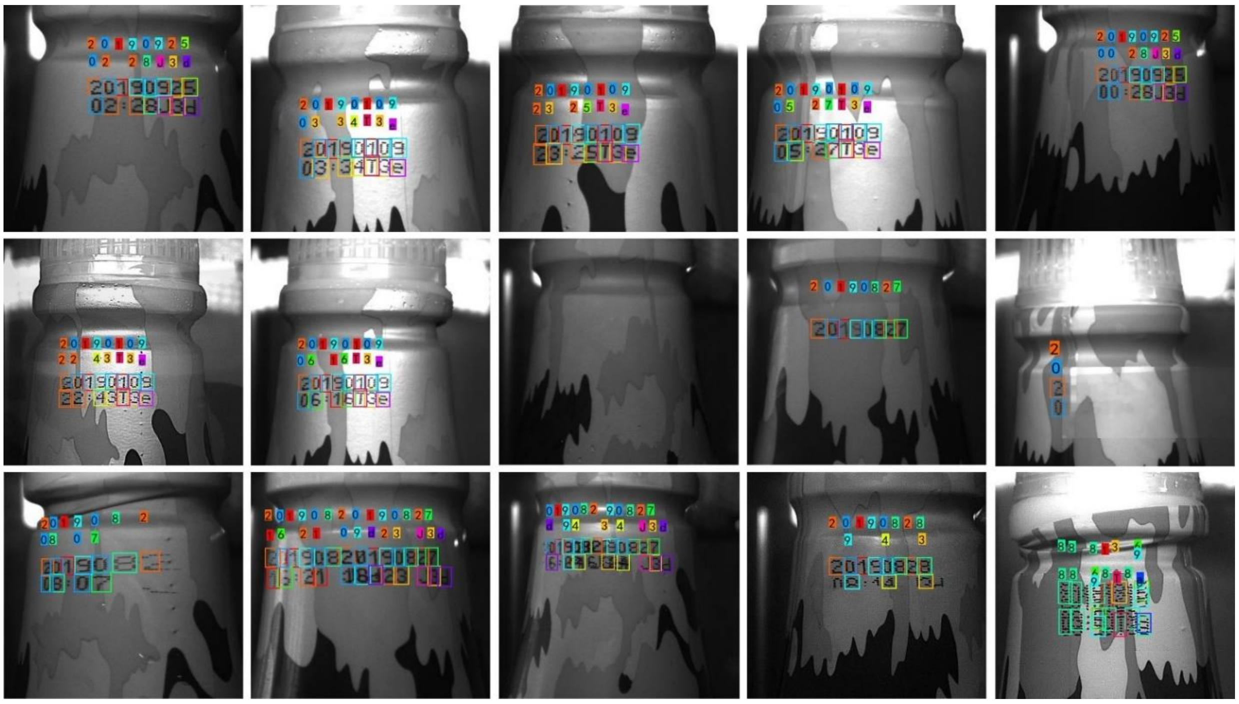
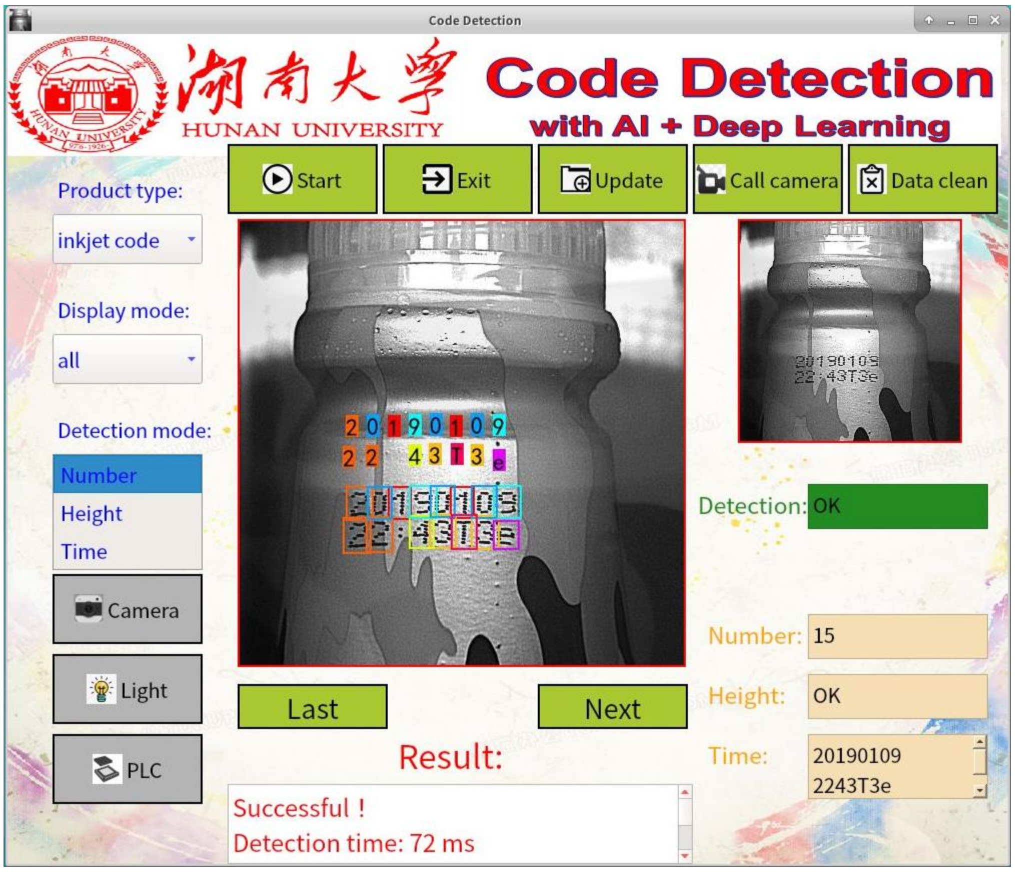


Figure 15. Partial detection results of the code images.

After the detection results have been obtained, the visual display becomes important for the quality inspection. Figure 16 demonstrates the interactive main interface of the code defect detection system developed by Python and PyQt5, which visually displays the detection results. A data set containing defective samples has been used for the experimental testing. Specifically, this data set contains a total of 675 code pictures, of which 175 are defective samples. Table 9 shows the experimental results of code detection, the numbers of miss detection and false detection are 1 and 0, respectively, which indicates a miss detection rate of 0.15% and an accuracy of 0.9985. Additionally, the average time consumption is around 72 ms, which is generally below the industrial time requirement of 100 ms. As clearly illustrated in the experimental results, the proposed network meets the requirement of the detection accuracy and speed for actual industrial production.

Table 9. Experimental results of code detection.

Class	Validation set	Miss detection	False detection	Accuracy	Speed
Normal	500	0	0	100%	72 ms
All loss	25	0	0	100%	72 ms
Partial loss	100	1	0	99%	72 ms
Duplication	25	0	0	100%	72 ms
Illegibility	15	0	0	100%	72 ms
Bonding	10	0	0	100%	72 ms
Total	675	1	0	99.85%	/



Product type: inkjet code

Display mode: all

Detection mode: Number

Height

Time

Detection: OK

Number: 15

Height: OK

Time: 20190109
2243T3e

Result:

Successful !
Detection time: 72 ms

Figure 16. The interactive main interface of the code defect detection system.

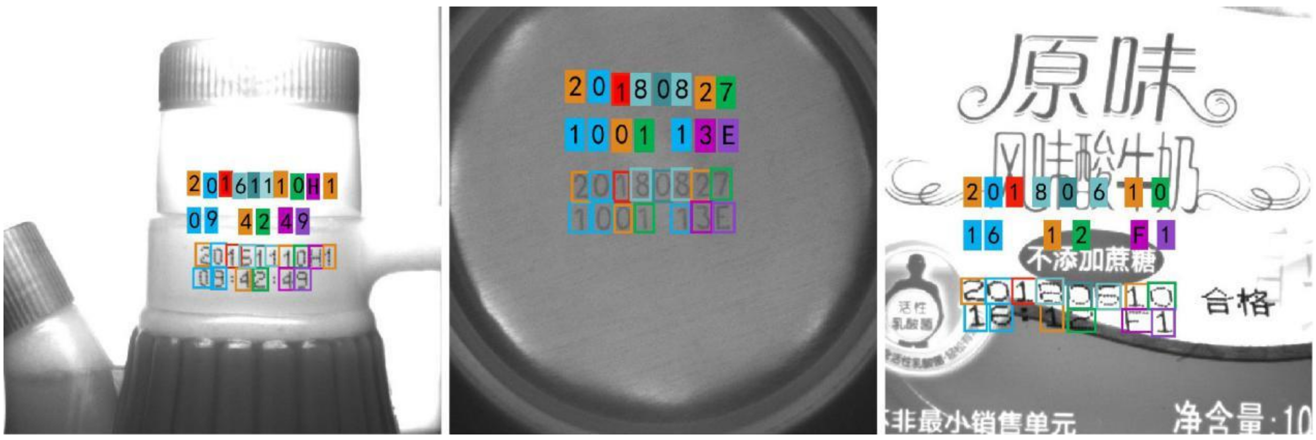
5. Transfer learning

Deep learning is a kind of supervised learning, which generally requires a large amount of data and label information to learn and train a high-performance model with a limited generalization ability for different test data sets. However, unlike the normal supervised learning, deep learning features the transfer learning, which can be used to solve the problems of super small training data set and poor generalization ability on different test sets [50]. Basically, transfer learning applies the existing knowledge to unlearned tasks, and it mainly include instances-based transfer learning, mapping-based transfer learning, network-based transfer learning and adversarial-based transfer learning. In this paper, the network-based transfer learning has been used to test the performance of the proposed detection method. Specifically, the detection network structure and some parameters were used in other training data sets for training the detection model and testing the detection performance.

Figure 17 shows the typical experimental code character detection results of transfer learning on the soy sauce bottles, the beer cans, and the milk packages. For these three kinds of packages, the original data sets contain 50, 50 and 46 images respectively, and the corresponding training data sets with simple data enhancement contain 195, 201 and 160 images respectively, which are all the small data sets. Table 10 demonstrates the experimental results of the transfer learning on the three packages. As clearly shown in the experimental results, the proposed detection network in this paper has strong feature extraction capabilities and it can easily adapt to different detection task scenarios with code characters on simple or complex backgrounds.

Table 10. Experimental results of transfer learning.

Detection object	Validation set	Miss detection	False detection	Accuracy
Soy sauce bottle	195	0	0	100%
Bear can	197	0	0	100%
Milk package	203	0	1	99.51%



(a) (b) (c)

Figure 17. The detection results of code characters from transfer learning on, (a) the soy sauce bottle, (b) the bear can, and (c) the milk package.

6. Conclusions

To sum up, an efficient deep learning-based defect detection method for code characters on complex backgrounds has been proposed. For the data preprocessing in this paper, the morphological operations were used to generate a large number of defective samples to balance positive and negative samples, and then the data enhancement technologies were implemented to further expand the data set. Also, the core module of EfficientNet was applied in the proposed detection method called BBE to construct the efficient feature extraction network, and then the further feature fusion was performed on the extracted features. Afterwards, relevant quality inspection standards were set to screen the detection results of the code characters to evaluate the overall quality of the code image. To verify the effectiveness of the proposed defect detection method, experiments have been conducted and the results showed a detection accuracy of 0.9985 and an average detection time of 72 ms for a single code image, which meet the requirements of the industrial production. Compared to other similar methods, the proposed method could exhibit better detection accuracy (0.9985) and faster speed (50000 bottles/hour) for the code characters on the complex background. Remarkably, it also be easily applied to similar detection tasks by transfer learning, with high robustness and generalization ability. This proposed method could provide new insights into the applications of the code character defection detection in the industrial production. In the future research, the attention will be mainly focused on the further improvement of the location accuracy of code characters and reducing the number of the code images for the detection model training.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (NSFC 62073129), the Chang-Zhu-Tan National Indigenous Innovation Demonstration Zone Project (2017XK2102), and the Nature Science Research Project of Anhui Province (1808085QF195).

Conflict of interest

The authors declare that there are no conflicts of interest related to this paper.

References

1. Y. Qin, Z. Zhang, Summary of scene text detection and recognition, in *2020 15th IEEE Conference on Industrial Electronics and Applications*, (2020), 85–89.
2. L. Shufeng, S. Shaohong, S. Zhiyuan, Research on Chinese characters recognition in complex background images, in *2017 2nd International Conference on Image, Vision and Computing (ICIVC)*, (2017), 214–217.
3. B. Wang, C. L. P. Chen, License plate character segmentation using key character location and projection analysis, in *2018 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC)*, (2018), 510–514.
4. R. A. P. Putro, F. P. Putri, M. I. Prasetiyowati, A combined edge detection analysis and clustering based approach for real time text detection, in *2019 5th International Conference on New Media Studies (CONMEDIA)*, (2019), 59–62.
5. V. V. Rampurkar, S. K. Shah, G. J. Chhajed, S. K. Biswash, An approach towards text detection from complex images using morphological techniques, in *2018 2nd International Conference on Inventive Systems and Control (ICISC)*, (2018), 969–973.
6. B. Tan, Q. Peng, X. Yao, C. Hu, Z. Xu, Character recognition based on corner detection and convolution neural network, in *2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC–FSKD)*, (2017), 503–507.
7. T. Zheng, X. Wang, X. Yuan, S. Wang, A Novel Method based on Character Segmentation for Slant Chinese Screen-render Text Detection and Recognition, in *2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, (2020), 950–954.
8. R. Bagi, T. Dutta, H. P. Gupta, Cluttered textspotter: An end-to-end trainable light-weight scene text spotter for cluttered environment, *IEEE Access*, **8** (2020), 111433–111447.
9. S. Y. Arafat, M. J. Iqbal, Urdu-text detection and recognition in natural scene images using deep learning, *IEEE Access*, **8** (2020), 96787–96803.
10. Y. S. Chernyshova, A. V. Sheshkus, V. V. Arlazarov, Two-step CNN framework for text line recognition in camera-captured images, *IEEE Access*, **8** (2020), 32587–32600.
11. L. Cao, H. Li, R. Xie, J. Zhu, A text detection algorithm for image of student exercises based on CTPN and enhanced YOLOv3, *IEEE Access*, **8** (2020), 176924–176934.
12. Z. Tian, W. Huang, T. He, P. He, Y. Qiao, Detecting text in natural image with connectionist text proposal network, in *European conference on computer vision*, (2016), 56–72.
13. X. Ren, Y. Zhou, Z. Huang, J. Sun, X. Yang, K. Chen, A novel text structure feature extractor for Chinese scene text detection and recognition, *IEEE Access*, **5** (2017), 3193–3204.

14. R. C. Gonzalez, R. E.Woods, Q. Ruan, Y. Ruan, *Digital Image Processing*, 3rd edition, China edition, Publishing House of Electronics Industry, Beijing, (2017), 404–411.
15. Z. Zou, Z. Shi, Y. Guo, J. Ye, Object detection in 20 years: A survey, preprint, arXiv:1905.05055.
16. Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, *Proc. IEEE*, **86** (1998), 2278–2324.
17. A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, *Commun. ACM*, **60** (2017), 84–90.
18. K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, preprint, arXiv:1409.1556.
19. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, et al., Going deeper with convolutions, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, (2015), 1–9.
20. K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, (2016), 770–778.
21. F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, K. Keutzer, SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5MB model size, preprint, arXiv:1602.07360.
22. F. Chollet, Xception: Deep learning with depthwise separable convolutions, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, (2017), 1251–1258.
23. A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, et al., Mobilenets: Efficient convolutional neural networks for mobile vision applications, preprint, arXiv:1704.04861.
24. Y. Ma, M. Huang, Study of digit recognition based on BP neural network, *Inf. Technol.*, **4** (2007), 87–91.
25. R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, (2014), 580–587.
26. K. He, X. Zhang, S. Ren, J. Sun, Spatial pyramid pooling in deep convolutional networks for visual recognition, *IEEE TPAMI*, **37** (2015), 1904–1916.
27. R. Girshick, Fast r-cnn, in *Proceedings of the IEEE international conference on computer vision*, (2015), 1440–1448.
28. S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, preprint, arXiv:1506.01497.
29. K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask r-cnn, in *Proceedings of the IEEE international conference on computer vision*, (2017), 2961–2969.
30. J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, (2016), 779–788.
31. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, et al., Ssd: Single shot multibox detector, in *European conference on computer vision*, (2016), 21–37.
32. M. Tan, Q. V. Le, Efficientnet: Rethinking model scaling for convolutional neural networks, preprint, arXiv:1905.11946.
33. P. Ramachandran, B. Zoph, Q. V. Le, Searching for activation functions, preprint, arXiv:1710.05941.

34. X. Y. Yin, J. Goudriaan, E. A. Lantinga, J. Vos, H. J. Spiertz, A flexible sigmoid function of determinate growth, *Ann. Bot.*, **91** (2003), 361–371.
35. X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, (2011), 315–323.
36. S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, preprint, arXiv:1502.03167.
37. M. Tan, R. Pang, Q. V. Le, Efficientdet: Scalable and efficient object detection, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2020), 10781–10790.
38. M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, et al., Tensorflow: A system for large-scale machine learning, in *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, (2016), 265–283.
39. D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, preprint, arXiv:1412.6980.
40. L. Bottou, Large-scale machine learning with stochastic gradient descent, in *Proceedings of COMPSTAT'2010. Physica–Verlag HD*, (2010), 177–186.
41. J. Redmon, A. Farhadi, Yolov3: An incremental improvement, preprint, arXiv:1804.02767.
42. Z. Jiang, L. Zhao, S. Li, Y. Jia, Real-time object detection method based on improved YOLOv4-tiny, preprint, arXiv:2011.04244.
43. *J. Glenn*, Yolov5, 2021. Available from: https://github.com/ultralytics/yolov5.
44. X. Zhou, D. Wang, P. Krähenbühl, Objects as points, preprint, arXiv:1904.07850.
45. N. Cai, Y. Chen, G. Liu, G. Cen, H. Wang, X. Chen, A vision-based character inspection system for tire mold, *Assem. Autom.*, **37** (2017), 230–237.
46. X. Yang, J. Zhang, Q. Zhou, A polynomial self-optimizing method for the character correction of distortion prints, *Packag. Eng.*, **39** (2018), 187–193.
47. F. Lei, K. Sun, X. Wang, Research on milk production date recognition based on improved LeNet-5, *Comput. Technol. Dev.*, **30** (2020), 96–99.
48. Y. Zhou, T. Shen, Z. Xi, Y. Yang, Z. Qiu, Research on quality detection algorithms based on halcon for bottom spray code of daily chemical bottles, *Instrum. Tech. Sens.*, **11** (2020), 101–104.
49. H. Wang, L. Zhu, Z. Pan, Method for the detection of the piston side defect based on external contour registration, *J. XiDian Univ.*, **46** (2019), 75–83.
50. C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, C. Liu, A survey on deep transfer learning, in *International conference on artificial neural networks*, (2018), 270–279.