Journal of Computers Vol. 35 No. 6, December 2024, pp. 185-199
doi: 10.53106/199115992024123506014

Attention-Based Lightweight Network for PCB Defect Data Augmentation

Hong-Yan Zhang[1], Xin Wang[1], and Jian-Wei Zhao[2*]

[1] College of Electrical and Control Engineering, Shenyang Jianzhu University,
Shenyang 110168, China,
799509995@qq.com, wangx7988@sjzu.edu.cn

[2] Department of Computer Information Engineering, Baoding Vocational and Technical College,
Baoding 071052, China
2238552865@qq.com

Received 29 January 2024; Revised 26 June 2024; Accepted 8 August 2024

Abstract. Performance of deep learning-based PCB (Printed Circuit Board) surface defect detection networks is often limited by the depth of feature extraction networks and the quality of training data. While significantly increasing network parameters can only slightly enhance system performance, optimizing training data can improve network performance without adding computational overhead. Therefore, this paper proposes a data augmentation network to enhance detection accuracy. First, an autoencoder generator is designed to enhance the feature fitting capability of the latent space. Second, a generative adversarial structural loss function is introduced, and an adversarial training method with different learning rates for the generator and discriminator is employed. Finally, experimental results demonstrate that this method enhances the diversity of PCB defect data and effectively improves the detection network's recognition accuracy.

Keywords: PCB surface defect detection, deep learning, data augmentation, small object detection

1 Introduction

With the rapid advancement of Integrated Circuit (IC) packaging technology, electronic devices are becoming increasingly thin and light, leading to tighter routing spaces on printed circuit boards (PCBs). Consequently, the quality requirements for PCBs have also increased. To ensure the high performance of electronic devices, PCB defect detection has become a critical technology in modern electronic product manufacturing. In recent years, the rapid development of deep learning has significantly improved the efficiency and accuracy of defect detection for high-precision devices like PCBs. However, the accuracy of these methods depends heavily on the training data. Due to time and cost constraints, situations often arise where there are few images, poor image quality, and imbalanced categories [1], posing numerous challenges for image recognition tasks.

Generative Adversarial Networks (GANs) offer a novel approach to object detection tasks by learning mappings from latent space to real distributions. Once the target distribution for a task is obtained, GANs can learn ways to "approximate the correct answer," pursuing a Nash equilibrium in the non-convex game of high-dimensional continuous parameters between the generator and the discriminator. Therefore, compared to traditional object detection training methods that learn fitting patterns from labeled data, GANs can be used in object detection to generate features, providing robustness for detecting objects in degraded images. However, GANs usually use gradient descent to process the loss of the generator and discriminator, leading to local minima of artificially designed loss functions rather than the Nash equilibrium point of the non-convex game. To address this issue, Salmans et al. proposed a feature matching method for generating images and seeking Nash equilibrium and a semi-supervised training method for object detection [2]. These methods add a category of generated images for training the object detector, which must classify real samples while adversarially interacting with the generator. However, while the discriminator should decide with a 1/2 probability whether the generated data is fake, the detector should reasonably classify the generated data into real categories. The balance between the discriminator's identification of fake samples and the detector's label prediction is difficult to achieve as the optimal point for network training. Therefore, applying detection and generation within a single framework is challenging. To ad-

* Corresponding Author

dress this issue, Li et al. [3-5] proposed a three-player game framework called Triple GAN, introducing a detector as a third party in the game between the generator and the discriminator. Despite careful design of the adversarial training process, the discriminator's negation of the detector's classification results for real images, along with the strong coupling between the generator and the detector, can hinder the detector's performance. Thus, extending the two-player game to three players is not conducive to stable training. In this context, we propose a novel approach that combines a well-trained autoencoder with a discriminator's feature extraction network for joint training to assess feature quality and optimize the performance of the feature extraction network. After training, both generated and original samples are fed into the detection network for training. Based on this, we propose an attention mechanism-based adversarial generator model, designing a lightweight data augmentation method for object detection. This continues to improve the generator network's performance, achieving better generation results with less training time, thereby enhancing the detection network's training and detection capabilities. The main contributions of this paper are as follows:

1) By designing an autoencoder generator, we enhance the latent space's feature fitting capability, allowing the feature extraction network to achieve additional improvements with lower training costs.

2) We propose a structural loss function that improves the network's perception of small object structures, enhancing reconstruction accuracy by emphasizing image boundaries and details.

3) We apply the proposed algorithm to augment three PCB small object defect datasets and validate it using mainstream detection networks. Experimental results show significant improvements in various detectors, particularly in detecting small object defects.

The structure of this paper is arranged as follows: first, we review related work pertinent to the proposed data augmentation algorithm in the related work section. Then, we provide a detailed introduction to the proposed model's structure. Next, in the experimental evaluation section, we introduce the datasets and experimental settings used, followed by an analysis and discussion of the results. Finally, the conclusion section summarizes the paper and discusses future research directions.

2 Related Work

2.1 Generative Adversarial Networks

Generative Adversarial Networks (GAN) create a minimax game between the generator and discriminator, with the objective function as follows:

$$L_{GAN} = V(D,G) = E_{x\sim P_{data}}\left[\log D(x)\right] + E_{z\sim P_z(z)}[\log(1-D(G(z)))]. \quad \textbf{(1)}$$

Equation (1) shows that GANs have two loss functions: log D(x) and log (1 - D(G(z))). The former is optimized for the discriminator, while the latter is optimized for the generator. When one component is trained, the parameters of the other remain fixed. Despite GANs' impressive performance in tasks like image generation and data augmentation, their training is highly unstable, suffering from issues like gradient vanishing and mode collapse. To mitigate the excessive freedom in GAN models, a natural approach is to impose constraints, leading to the introduction of Conditional Generative Adversarial Nets (CGAN) [6]. CGANs guide data generation by incorporating additional information conditions y. If y is a class label, CGANs can transform an unsupervised GAN into a supervised model. This straightforward enhancement has proven highly effective and is widely used in subsequent related work [7]. Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets (InfoGAN) [8] is another significant model in the realm of conditional GANs. InfoGAN introduces an additional latent code c in the generator alongside noise z. The term "Info" denotes mutual information, indicating the relationship between the generated data x and the latent code c. To enhance the relationship between x and c, the value of Info must be maximized, which effectively adds a mutual information regularization term. By varying the latent code c, InfoGAN can control attributes of the generated images, such as the tilt or thickness of digits and the rotation of 3D models of faces [9]. Although these methods have improved GANs to

some extent, they have not completely solved the issues of training instability and gradient vanishing. To address the gradient vanishing problem during training, Wasserstein Generative Adversarial Networks (WGAN) introduced a novel approach. By providing the network with a simple gradient (adding 1 if the output is real, and subtracting 1 otherwise), WGAN minimizes the Wasserstein distance (also known as the "earth mover's" distance) [10, 11]. The main contribution of WGAN is stabilizing the training process, thereby avoiding the common gradient vanishing problem seen in traditional GANs. Boundary Equilibrium Generative Adversarial Networks (BEGAN) use an autoencoder as the discriminator, creating a new loss function derived from the Wasserstein distance and the reconstruction loss of real and generated images [12]. Sultana et al. [13] developed a model with three structures: generator, detector, and feature extraction network. Despite having a module specifically designed to capture gradient information, its performance is still constrained by the feature distribution of the original images.

2.2 Object Detection Task

With the advancement of deep learning, particularly the successful application of Convolutional Neural Networks (CNNs), the field of object detection has undergone significant changes. In 2015, Redmon et al. introduced the YOLO (You Only Look Once) single-stage algorithm for the first time. The core idea of YOLO is to transform the object detection problem into a single regression problem, directly mapping from image pixels to bounding box coordinates and class probabilities [14]. Unlike traditional two-stage methods such as the R-CNN series [15, 19], single-stage algorithms do not require generating candidate regions; they directly predict the class and position of objects from the input image, thereby improving detection speed. However, they perform poorly in detecting small objects. To address this issue, Young-Jin Cha et al. improved the CNN architecture, enhancing the detection accuracy for small objects to 98%, though it takes 4.55 seconds to process a single image [13]. To balance accuracy and speed, SSD introduced multi-scale feature maps, which improved the detection of small objects [16]. RetinaNet proposed a focal loss to address the common issue of foreground-background class imbalance in object detection and used Feature Pyramid Networks (FPN) to efficiently detect objects of different scales [21, 22]. FasterNet-SSD replaced the original SSD model's VGG16 backbone with the FasterNet network based on Partial Convolution (PConv), achieving an average precision (mAP) of 80.38% on the small object test set, though at the cost of reduced detection speed [20]. YOLOv7 improved network architecture and training efficiency, enabling faster training on smaller datasets and achieving an mAP of 87.51% on the small object test set, while reducing computational requirements by 36% compared to the previous version. However, it still struggles with accuracy in multi-scale object detection scenarios [17]. Recently, Reis, D. et al. employed anchor-free detection and online data augmentation techniques in YOLOv8, enhancing inference speed and training efficiency. However, due to the nature of convolution operations, the network still fails to meet industrial requirements for the detection accuracy of tiny objects [18]. Unlike the aforementioned methods, our approach does not involve modifying a specific network architecture. Instead, it employs a "preprocessing" technique to optimize training data, allowing it to coexist with other techniques that improve feature quality.

3 Lightweight Network Framework

The lightweight network framework proposed in this paper is illustrated in Fig. 1. Our experimental dataset comprises publicly available DEEPPCB and PCBDATASET, as well as industrial datasets collected independently. In the image reconstruction phase, we first construct a compact and efficient network structure to reduce computational resource demands while fully extracting and reconstructing data features, thereby enhancing network operational efficiency. Secondly, we employ an adversarial reconstruction loss function to transform the standard error evaluation method into a more refined and structure-aware assessment system. Finally, a multi-level loss function is designed, and different learning rates are set for the generator and adversarial networks to improve overall network stability. Additionally, the introduction of the Discriminator network enhances the generative capabilities of the Generator network. The interaction and optimization strategies between the Lightweight generative network structure and Discriminator networks jointly contribute to performance improvement.

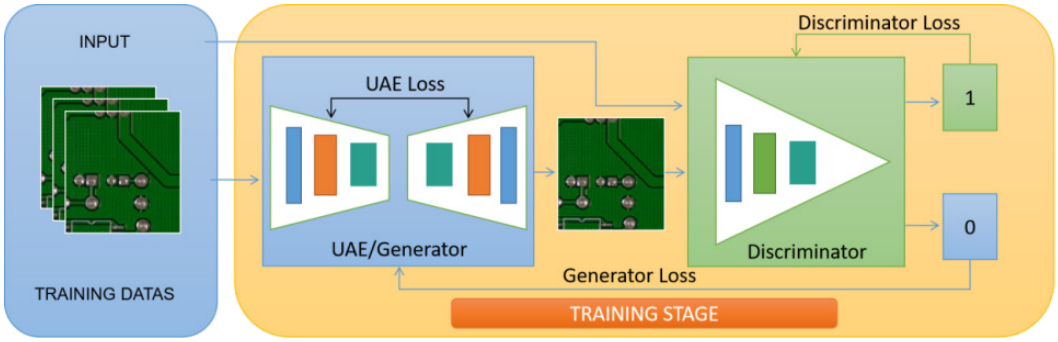

Fig. 1. Lightweight network overall framework

3.1 Improved Refactoring Network

In the research field of PCB surface defect detection, the challenge of detecting tiny targets is particularly prominent. Defects such as mouse_bite are especially difficult to detect due to their small size and their tendency to blend with the background color. Furthermore, the area difference between various defects can exceed fivefold, significantly increasing the difficulty of feature extraction. Existing generative networks struggle with these small defects because features may be lost during the forward propagation in clustering networks, thereby affecting reconstruction quality. To address this issue, this paper proposes an optimized structure for the extraction and reconstruction of small target defect features on PCB surfaces. The structure of this network is shown in Fig. 2, comprising two parts: the Encode section for feature extraction and the Decode section for image reconstruction.

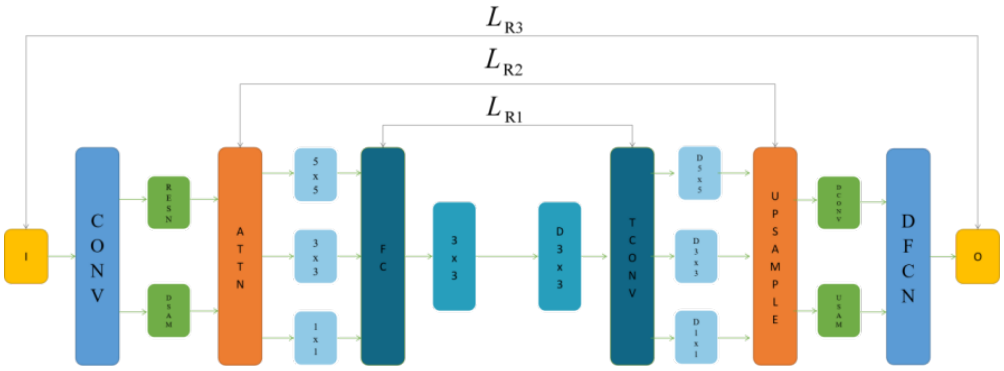

Fig. 2. Lightweight generative network structure

First, a convolution operation with a stride of 2 is applied to subdivide the feature map into four parts, which are then fed into two different branch networks to further extract feature information at various scales. Using the ResNet18 network structure, skip connections are employed to address the vanishing or exploding gradient problems in deep neural networks. Subsequently, a hybrid attention module (ATTN) is utilized to distinguish the importance between different channels, emphasizing critical information in the shallow network while suppressing irrelevant information. By employing convolutional kernels of different sizes, the model can simultaneously capture multi-scale information, thereby achieving a more comprehensive understanding and representation of the input data. Larger convolutional kernels or feature extraction layers help capture broader contextual information to understand the global structure and semantics, while smaller convolutional kernels or feature extraction layers help capture finer details and local structures. Fully connected layers are used to capture nonlinear patterns and feature correlations in the data, and the parameter-sharing mechanism helps reduce the risk of overfitting. Finally,

in the decoder network, a symmetric deconvolution structure is used to restore the image, enhancing the spatial dimensions and spatial resolution of the input data.

3.2 Network Structure

This paper employs an autoencoder (AE) as the core architecture for both the generator and discriminator, and introduces a hybrid module as an intermediate structure. The Discriminator network enhances the resolution of generated images through adversarial training. For detailed network architectures, please refer to Table 1 and Table 2.

Table 1. Improved reconstruction network

Network	Hierarchy	Input	Output
Encode	Conv BN Leaky Relu	(256, 256, 3)	(256, 256, 64)
	Conv BN Leaky Relu	(256, 256, 64)	(128, 128, 128)
	Conv BN Leaky Relu	(128, 128, 128)	(64, 64, 256)
	ATT-LAYER	(64, 64, 256)	(32, 32, 512)
	ATT-LAYER	(32, 32, 512)	(32, 32, 512)
	Conv BN Leaky Relu	(32, 32, 512)	(32, 32, 512)
	Conv BN Leaky Relu	(32, 32, 512)	(32, 32, 512)
	Conv BN Leaky Relu	(32, 32, 512)	(32, 32, 512)
Decode	DEConv BN Leaky Relu	(32, 32, 512)	(32, 32, 512)
	DEConv BN Leaky Relu	(32, 32, 512)	(32, 32, 512)
	DEConv BN Leaky Relu	(32, 32, 512)	(64, 64, 256)
	DEConv BN Leaky Relu	(64, 64, 256)	(128, 128, 128)
	DEConv BN Leaky Relu	(128, 128, 128)	(256, 256, 64)
	DEConv BN Leaky Relu	(256, 256, 64)	(256, 256, 3)

Table 2. Discriminator network architecture

Network	Hierarchy	Input	Output
Input layer	Conv-Leaky Relu	(256, 256, 3)	(256, 256, 64)
	Conv layer	(256, 256, 64)	(256, 256, 64)
	Conv layer	(256, 256, 64)	(128, 128, 128)
	Conv layer	(128, 128, 128)	(64, 64, 256)
Inner layer	Hidden layer	(64, 64, 256)	(32, 32, 512)
	Hidden layer	(32, 32, 512)	(32, 32, 64)
	Deconv layer	(32, 32, 64)	(64, 64, 64)
	Deconv layer	(64, 64, 64)	(128, 128, 64)
	Deconv layer	(128, 128, 64)	(128, 128, 64)
	Deconv layer	(128, 128, 64)	(256, 256, 64)
Output layer	Conv-Leaky Relu Conv-Leaky Relu Conv-Tanh	(64, 64, 256)	(256, 256, 3)

The Autoencoder (AE), as a network structure that connects the encoder and decoder, allows more original image texture information to propagate through the high-level feature layers. For the generator, the first convolutional layer of the encoder uses 64 filters, while the last convolutional layer of the decoder is responsible for mapping the number of channels to that of the output image (3 channels for color images and 1 channel for black-and-white images). Apart from the first convolutional layer of the encoder, all convolutional layers are followed by batch normalization layers to standardize the features.

3.3 Adversarial Reconstruction Loss Function

In traditional networks, a single adversarial loss function is typically used to evaluate the convergence of the model, which to some extent loses structural information in the images. To reduce the impact of the loss function on the fine-grained structure of Printed Circuit Board (PCB) defects, a structural loss is employed to enhance the structural features of defects and facilitate network convergence.

1) Structural Loss Function

When assessing the performance of the reconstruction network model, the L1 loss and L2 loss are used to compare differences pixel by pixel, without considering the regional features of the images. For the detection of non-regular texture images, a Structural Similarity Index Measure (SSIM) is introduced to construct the loss function, enabling the network to handle complex and diverse texture background images and produce better results. The SSIM loss function optimizes the model based on three metrics: brightness, contrast, and structure, making it more capable of capturing image details compared to L1 and L2 loss functions. For input and output images of the model (x, y), SSIM is defined as:

$$SSIM\left(X,Y\right)=\left(\left(l\left(x,y\right)\right)^{\alpha}\left(c\left(x,y\right)\right)^{\beta}\left(s\left(x,y\right)\right)^{\gamma}\right). \tag{2}$$

$$l\left(x,y\right)=\frac{2\mu_x\mu_y+C_1}{\mu_x^2+\mu_y^2+C_1}. \tag{3}$$

$$\mathrm{s}\left(x,y\right)=\frac{\sigma_{xy}+C_3}{\sigma_x\sigma_y+C_3}. \tag{4}$$

In the equation: $\alpha>0,\beta>0,\gamma>0$, l(x, y) represents brightness comparison, c(x, y) represents contrast comparison, s(x, y) represents structure comparison. μ_{x} and μ_{y} respectively yield the mean values of x and y. σ_{x} and σ_{y} respectively determine the standard deviations of x and y. σ_{xy} is the covariance of x and y. All of C_1,C_2,C_3 are nonzero constants used to maintain the stability of l(x, y), c(x, y), and s(x, y).

SSIM loss function is defined as:

$$L_{SSIM}\left(x,y\right)=1-SSIM\left(x,y\right). \tag{5}$$

Reconstructing the network, the SSIM loss function is used to evaluate the difference between the decoder output layer and the original image. It is also employed to extract multiple different-scale deconvolution results and corresponding convolution layer results simultaneously using SSIM, thus constructing Multi-scale SSIM (MSSIM). For M scales, the MSSIM loss function is defined as:

$$L_{MSSIM}=1-\prod SSIM\left(x,y\right). \tag{6}$$

2) Adversarial Reconstruction Loss Function

Compared to the L2 loss, the L1 loss imposes a weaker penalty on pixel-level errors, making it suitable for irregular textured images. L_{MSSIM} can train the reconstruction network to focus on changes and color deviations in the samples, and by extracting corresponding results at multiple different scales, it preserves the edges and details of the image. In order to simultaneously address the defect detection problem in both regular and irregular texture samples and generate reconstruction images that are more similar to the original image, this paper proposes an adversarial reconstruction loss function. The combination of Loss L_{MSSIM} and Loss L_{D} serves as the loss function for the lightweight data generation network model, as follows:

$$L_{MSD} = \alpha\left(L_1 + L_{MSSIM}\right) + \left(1-\alpha\right)L_D \,. \tag{7}$$

In the equation, α represents a weighting factor, which has a range of (0,1) and is used to balance the proportion of reconstruction loss and discriminator loss.

3.4 Discusses the Multi-level Calculation of the Loss Function

In the reconstruction of small defects on PCB surfaces, the focus is on enhancing the model's ability to recognize edges and details. This is achieved through multiple calculations of the loss function. Each calculation effectively propagates the error from the decoder back to the encoder, encouraging the encoder to learn more precise feature representations. During this process, prediction results at different levels capture boundary and detail information at various scales, significantly improving the model's ability to perceive and reconstruct these critical features. Early in the training, the network may exhibit large prediction errors, but by repeatedly calculating the loss function, the network receives more error signals, making it easier to correct these mistakes. As training progresses, prediction errors gradually decrease. This mechanism provides more supervisory signals, helping to prevent the model from overfitting (Fig. 3).

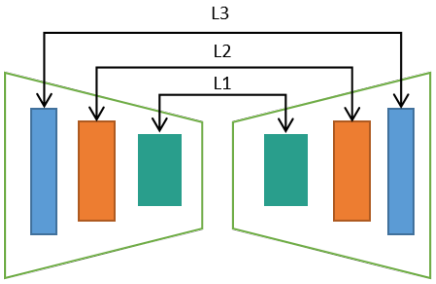

Fig. 3. Multi-level loss function

Slow convergence speed and mode collapse are common issues in generative networks. To address these problems, this paper designs an adversarial autoencoder network with a dual-rate updating rule. In this network, by setting a threshold for the change in the loss function, the network can determine whether the learning process is too fast or too slow and automatically adjust the update frequency accordingly. Additionally, different learning rates are assigned to the generator and the discriminator. Given that the discriminator needs to quickly adapt and distinguish between real and generated data, it is usually set with a higher learning rate. In contrast, the generator aims to produce increasingly realistic data, requiring a more meticulous and slower learning process; thus, it is assigned a lower learning rate.

4 Experiment

In this section, extensive experiments on both the public and self-built datasets of the proposed lightweight PCB defect data augmentation network based on the attention mechanism will be conducted. First, the datasets used in the experiments are introduced, followed by the key metrics for evaluating the model. Subsequently, the proposed algorithm is compared with other similar algorithms. All tests are conducted on a single computer, with the specific configuration shown in Table 3.

Table 3. Computer configuration table

Systems	WIN10
RAM	64GB
CPU	CORE I9-12900H
GPU	NVIDIA RTX3060
Deep learning framework	Pytorch, CUDA Version 11.8

4.1 Dataset Introduction

This experiment utilizes three datasets: the PCB-DATASET from Peking University, the Deeppcb dataset from Shanghai Jiao Tong University, and a self-constructed dataset. As shown in Fig. 4, the data originates from the PCB-DATASET, which consists of image data from an industrial production environment. This dataset contains 1,386 image samples covering six different PCB defect categories: spurious copper, spur, short, open circuit, mouse bite, and missing hole. As depicted in Fig. 4, the DEEPPCB dataset comprises 8,850 pairs of PCB images, including six types of defects: pin-hole, mouse bite, open, short, spur, and spurious copper. Each pair consists of a 640×640 defect-free template and a defective test image. The self-constructed (SELFPCB) dataset includes 750 PCB images, with each pair containing a 235×235 defect-free template and a defective test image.

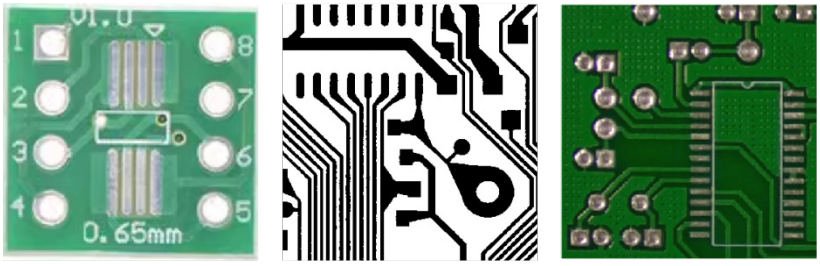

Fig. 4. PCB defect dataset

4.2 Model Evaluation Metrics

In the experiments, this paper utilizes pixel-level similarity metrics to assess the generated images. These include: Structural Similarity (SSIM), Peak Signal-to-Noise Ratio (PSNR), and Sharpness Difference.

1) Structural Similarity (SSIM) is a metric used to measure the similarity between two images, primarily based on the luminance, contrast, and structural information of the images. The formula for calculating SSIM is:

$$SSIM(I_g, I_v) = \frac{(2\mu I_g \mu I_v + c_1)(2\delta I_g I_v + c_2)}{(\mu^2 I_g + \mu^2 I_v + c_1)(\delta^2 I_g + \delta^2 I_v + c_2)}. \quad (8)$$

In this context, μ_{Ig} and μ_{Iv} respectively represent the mean values of the reconstructed image Ig and the real image Iv, while σIg and σIv are the standard deviations of the two images. Constants c_1 and c_2 are used to maintain stability.

2) Peak Signal-to-Noise Ratio (PSNR) is a metric used to assess image quality. It is calculated by comparing the maximum possible pixel values between the original and the distorted images with the mean squared error. The formula for calculation is as follows:

$$PSNR(I_{\mathrm{g}}, I_v) = 10\lg(\frac{\max^2 I_g}{mse(I_{\mathrm{g}}, I_v)}). \tag{9}$$

In this context,

$$mse(I_{\mathrm{g}}, I_v) = \frac{1}{\mathrm{n}}\sum_{i=1}^{n}(I_{\mathrm{g}}[i] - I_v[i])^2, \max_{I_g} = 255\max_{I_g}. \tag{10}$$

is the maximum pixel value in the image I_g, and mse represents the mean squared error.

3) Sharpness Discrepancy: SD is an index used to measure the sharpness discrepancy between synthetic images and real images. The formula for calculating SD is:

$$S\mathrm{harpDiff}(I_{\mathrm{g}}, I_{\mathrm{v}}) = 10\lg(\frac{\max^2 I_{\mathrm{g},}}{grads}). \tag{11}$$

$$grads = \frac{1}{N}\sum_i\sum_j(\mathrm{I}(\nabla_i I_v + \nabla_j I_v) - (\nabla_i I_g + \nabla_j I_g)\mathrm{I}). \tag{12}$$

In this context, 'grads' refers to the average value of image gradient differences. This is calculated by computing the absolute value of gradient differences for each pixel in the image and then taking the average.

4) This paper selects the average Intersection over Union (MIoU) threshold of 0.5 for the predicted bounding boxes compared to the target boxes. IoU represents the overlap rate between the detection result (DR) bounding boxes and the ground truth (GT) bounding boxes. The evaluation metrics for the model include Average Precision (AP), Mean Average Precision (mAP), and Frames Per Second (FPS). Precision (P) denotes the proportion of correctly predicted samples among all targets. AP is the area enclosed by the Precision-Recall (P-R) curve and the x-axis, and its calculation formula is as shown in the following figure:

$$P = \frac{TP}{TP + FP}. \tag{13}$$

$$R = \frac{TP}{TP + FN}. \tag{14}$$

$$AP = \int_0^1(P(R)\mathrm{d}R). \tag{15}$$

$$\mathrm{m}AP = \frac{\sum(AP)}{n} \tag{16}$$

$$I\mathrm{oU} = \frac{GT \cap DR}{GT \cup DR} \tag{17}$$

4.3 Analysis of Ablation Experiments

To comprehensively analyze the superiority of a lightweight PCB defect data augmentation network based on attention mechanisms, this paper designs ablation experiments on the foundation of a GAN network. The improved reconstruction network (A), adversarial reconstruction loss function (B), and multi-level computation loss function (C) were added separately for comparison with the original network. The specific experimental test results are shown in Table 8. By analyzing the contribution of each improvement strategy to the network as indicated in

Table 4, it was found that each module contributes to varying degrees of enhancement in the overall performance of the model.

Table 4. Results of ablation experiment

Model number	GAN	A	B	C	PCB-DATASET			DEEP-PCB			SLFE-PCB		
					SSIM	PSNR	SD	SSIM	PSNR	SD	SSIM	PSNR	SD
1	√				0.35	17.77	20.23	0.48	17.83	22.08	0.25	15.81	22.97
2	√	√			0.45	19.07	19.63	0.50	19.89	19.77	0.27	16.38	17.89
3	√		√		0.42	18.44	18.44	0.50	20.28	19.53	0.28	16.38	17.27
4	√			√	0.53	23.15	18.67	0.52	21.57	19.45	0.33	17.58	17.35
5	√	√	√	√	0.57	26.47	18.55	0.59	23.89	19.28	0.35	19.76	16.59

Based on the ablation experiment results presented in Table 4, it is evident that the improved reconstruction network achieves a PSNR improvement of over 1 in all three datasets. This indicates that the enhanced reconstruction network is better at aggregating image features. The introduction of the adversarial reconstruction loss function reduces structural reconstruction loss and more effectively guides the model to focus on the structural characteristics of small targets. The multi-level loss better compares the original and reconstructed images at different stages, ensuring that the generated images retain more of the original features. The results demonstrate that the lightweight data augmentation network can produce higher quality data.

4.4 Comparative Experiment on Adversarial Reconstruction Loss Function

To comprehensively evaluate the impact of loss functions on model performance, a comparative experiment was conducted using L_1+L_D, L_2+L_D, L_{MSSIM}+L_D, and L_{MSD}. The network parameters for this experiment are detailed in Table 5.

Table 5. Default network parameters

Sample size	3034 x 1586
Number of samples	1000
Number of iterations	1000
Loss function weight coefficient α	0.5

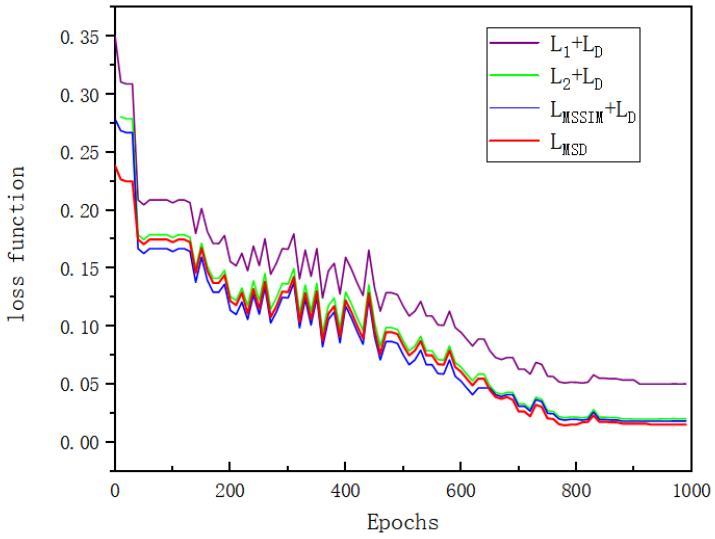

Fig. 5. Comparison of convergence curves for different loss functions

The convergence curves of the loss functions shown in Fig. 5 indicate that the L_1+L_D loss curve has poor convergence. The L_{MSD} loss function performs well in the first 600 iterations but surpasses L_{MSSIM}+L_D after 650 iterations. This improvement is attributed to the incorporation of a structural similarity factor and the introduction of a loss weight α in the loss function, allowing the model to better balance structural and generative losses. Consequently, the network adapts more effectively to images with complex and variable textures, leading to superior reconstruction results and the best convergence trend.

Table 6. Comparative experiments of different loss functions

Metrics	Sample	L1+ L_D	L2+ L_D	$L_{MSSIM}+L_D$	L_{MSD}
SSIM	PCB-DATASET	0.37	0.33	0.48	0.57
	Deeppcb	0.32	0.47	0.50	0.59
	SLFE-PCB	0.14	0.19	0.22	0.35
PSNR	PCB-DATASET	18.44	17.56	26.01	26.47
	Deeppcb	20.18	21.33	20.88	23.89
	SLFE-PCB	17.32	18.66	16.31	19.76
SD	PCB-DATASET	20.47	21.63	18.39	19.58
	Deeppcb	20.02	17.67	19.36	13.58
	SLFE-PCB	20.88	21.77	20.44	17.69

To provide a more comprehensive comparison of the impact of different loss functions on reconstruction performance, this study evaluated the performance metrics of the four loss functions across three different datasets. The results in Table 6 demonstrate that the L_{MSD} loss function outperforms the other three loss functions on samples from different datasets, confirming its excellent generalization capability across various scenarios.

4.5 Comparison with Similar Algorithms

The proposed model in this paper is compared with mainstream image generation and data augmentation models, including GAN, CycleGAN, VariGAN, and WGAN. Both subjective visual evaluation and objective parameter evaluation are used to demonstrate the effectiveness of the proposed model.

Subjective Evaluation. To verify the generalization capability of the improved model, this paper enhances low-light images in the PCB dataset and compares the results with traditional methods and our proposed method. The specific comparison results are shown in Fig. 6.

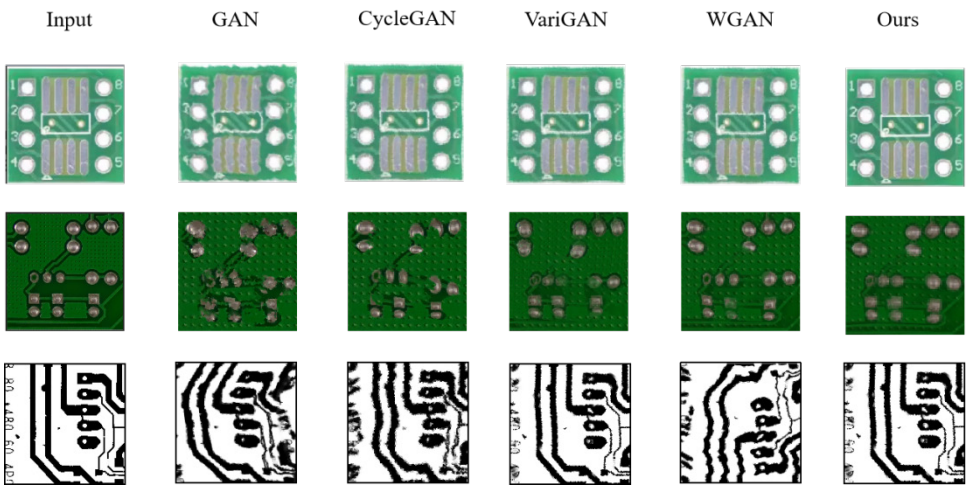

Fig. 6. Comparison of experimental results of similar algorithms

From the comparison images, it can be seen that the enhanced effect of the improved model on the PCB dataset is significantly better than that of traditional methods. Although other methods have some degree of image clarity restoration, there is still a considerable gap in brightness and structure compared to real images. In contrast, the improved model in this paper achieves better reconstruction effects in these aspects, providing a superior subjective visual experience. It is evident that our model excels in detail recovery and color restoration, offering higher visual quality and accuracy.

Objective Evaluation. To objectively verify the generalization capability of our algorithm across three different datasets, objective metrics are used to evaluate the generated images. The results are shown in Table 7.

Table 7. Objective evaluation comparison

Dataset	Metrics	GAN	CycleGAN	VariGAN	WGAN	Ours
PCB-DATA	SSIM	0.35	0.43	0.39	0.54	0.57
	SD	29.66	23.12	21.57	27.34	19.58
	PSNR	12.66	10.69	15.24	16.01	26.47
DEEP-PCB	SSIM	0.33	0.56	0.43	0.53	0.59
	SD	19.61	24.23	28.36	16.67	13.58
	PSNR	19.66	21.06	21.45	22.63	23.89
SELF-PCB	SSIM	0.29	0.15	0.24	0.27	0.35
	SD	39.46	38.14	27.11	34.61	17.69
	PSNR	14.67	14.88	16.61	16.01	19.76

In our experiments, we compared various GAN methods on the PCB-DATA, DEEP-PCB, and SELF-PCB datasets. The results indicate that our proposed method outperforms traditional methods in terms of Structural Similarity Index (SSIM), Standard Deviation (SD), and Peak Signal-to-Noise Ratio (PSNR). Specifically, our method achieved the highest SSIM (0.57) and PSNR (26.47), and the lowest SD (19.58) on the PCB-DATA dataset; on the DEEP-PCB dataset, SSIM reached 0.59, PSNR was 23.89, and SD was 13.58; on the SELF-PCB dataset, SSIM was 0.35, PSNR was 19.76, and SD was 17.69. By optimizing the generator structure, improving the loss function, and adjusting the training methods, we significantly enhanced the quality of the generated images and the stability of the model training, demonstrating superior generalization capability and visual effects. These improvements enable our generator to better capture the complex features of the data, producing more realistic images and showing good adaptability and stability across different datasets.

4.6 Comparison of Computational Load

To further verify the effectiveness of the lightweight algorithm, we compared the computational load of the lightweight data augmentation network based on the attention mechanism with the baseline model on the PCB-DATASET dataset. The experimental results are shown in Table 8.

Table 8. Comparison results of calculation quantity

	GAN	ClycleGAN	VariGAN	WGAN	Ours
Size	640	640	640	640	640
Params (M)	8.86	2.15	6.78	6.06	1.04
FLOPs (G)	5.62	3.5	6.96	5.98	2.7

As shown in Table 8, our proposed algorithm demonstrates significant advantages in terms of computational load and parameter count. Specifically, our algorithm's computational load is only 2.7G, and the number of parameters is only 1.04M, compared to the computational load of 5.62G to 6.96G and parameter count of 2.15M to 8.86M of other models, reducing a substantial amount of computational resources and storage requirements. The lower computational load means that the algorithm can process high-resolution input images more quickly and efficiently, while also reducing energy consumption, making it suitable for real-time image processing applications. Additionally, the reduction in the number of parameters helps simplify the model structure, improve train-

ing and adjustment efficiency, and reduce the risk of overfitting, thereby enhancing the model's generalization capability and deployment efficiency.

4.7 Impact of Data Augmentation on Object Detection Networks

In this experiment, YOLOv8 is selected as the baseline network, and training is conducted on the public PCB-DATASET for 200 iterations to comprehensively explain the impact of training data on the convergence performance of the network. The training result curves are shown in Fig. 7 and Fig. 8.

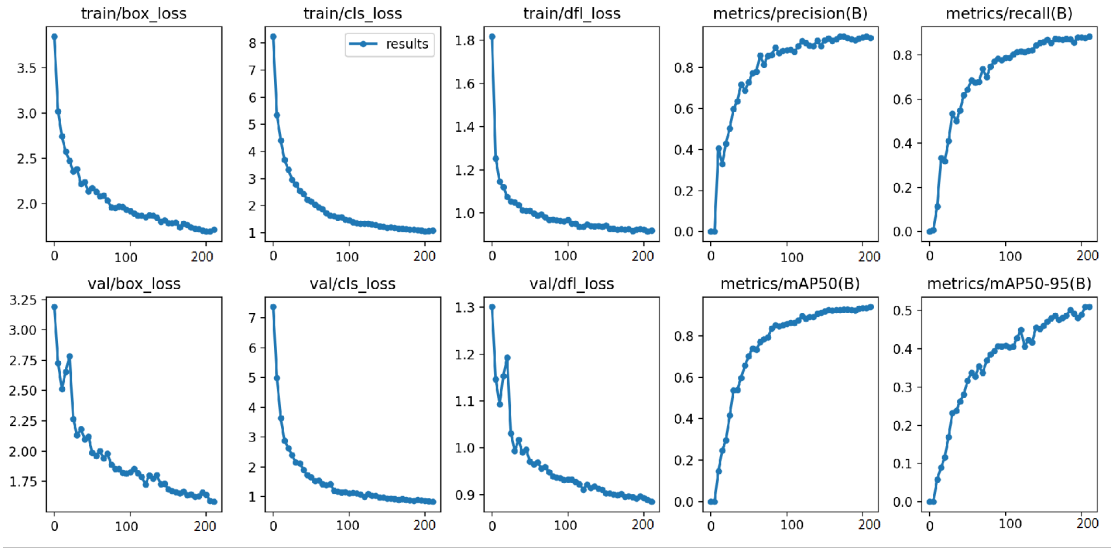

Fig. 7. Network training curve before data augmentation

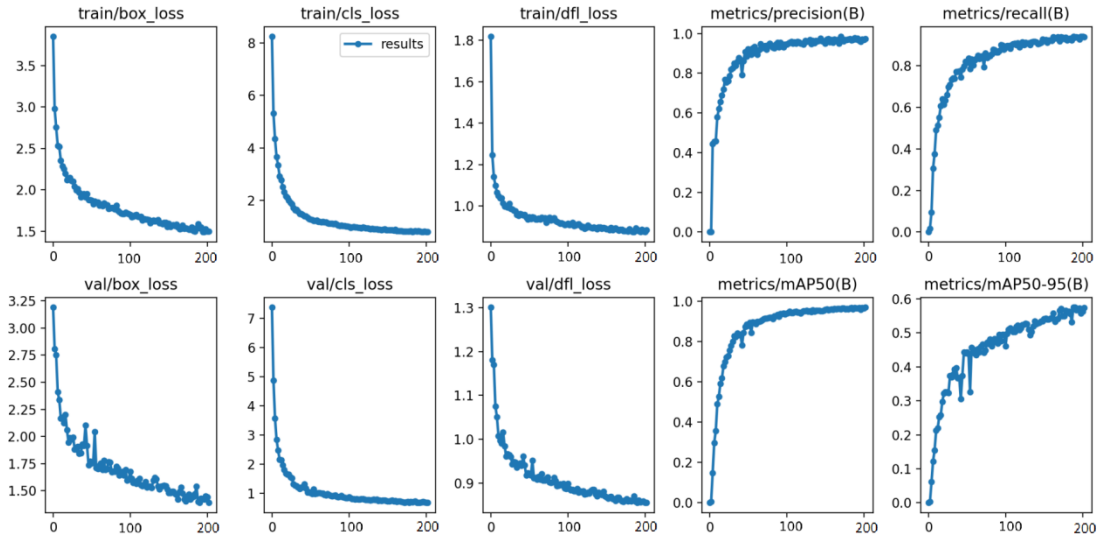

Fig. 8. Network training curve after data augmentation

The experimental results show that introducing sample variation effectively expands the diversity of the training dataset and reduces the model's dependence on specific data features. Data augmentation not only improves

the learning efficiency of the network but also accelerates the convergence process. As shown in Fig. 7, without data augmentation, the network training converges slowly, and there are significant fluctuations in the validation set loss, indicating suboptimal generalization ability for unknown data. In contrast, data augmentation forces the network to learn more generalized feature representations by continuously varying the presentation of samples. This not only reduces the model's dependence on single samples or features but also improves its robustness when facing new samples. In Fig. 8, the fluctuations in the validation set loss are reduced, fully proving this point. Under the same number of iterations, the model's mAP value also increases, demonstrating that the augmented data enables the network to achieve higher recognition accuracy.

To further verify the effectiveness of the proposed method, we compared the detection performance of the dataset before and after augmentation using classic algorithms in the defect detection field, including IPG-Net, Faster R-CNN, SSD, RFBNet, PFPNet, and YOLOV8. The comparison experiment used mAP@0.5 as the evaluation metric for each algorithm, and the detection results are listed in Table 9.

Table 9. Results of different detection algorithms

	SSD (mAP@0.5)	RFBNet (mAP@0.5)	PFPNet (mAP@0.5)	YOLOv8 (mAP@0.5)	Faster R-CNN (mAP@0.5)	IPG-Net (mAP@0.5)
PCB-DATASET	72.7	80.4	80.1	88.7	69.1	82.6
ENH-PCB-DATASET	75.6	81.2	81.7	90.7	74.3	84.5
DEEP-PCB	64.0	78.4	76.1	81.5	68.7	74.8
ENH-DEEP-PCB	64.4	81.0	79.8	86.1	71.7	79.2
SLFE-PCB	73.1	87.2	80.4	90.1	78.6	83.9
ENH-SLFE-PCB	78.1	87.2	82.6	91.4	79.5	86.8

As shown in Table 9, the experimental analysis reveals that the data augmentation method significantly improves the performance of various defect detection algorithms, especially when dealing with complex and diverse datasets. Specifically, algorithms such as YOLOv8, Faster R-CNN, and IPG-Net show a substantial increase in mAP@0.5 after data augmentation, indicating that they can fully utilize the augmented data to improve detection accuracy. The comprehensive results demonstrate that data augmentation not only enhances the model's ability to detect small targets but also significantly improves its generalization performance on complex datasets.

5 Conclusion

Given the diversity of surface defect types in Printed Circuit Boards (PCBs), the minute appearance of defects, and the time and cost constraints of image acquisition, this paper proposes a lightweight PCB defect data augmentation network based on an attention mechanism. This approach reduces the demand for computational resources, effectively extracts and reconstructs data features, and improves the efficiency of network operations. An adversarial reconstruction loss function was designed, employing a multi-level loss function and setting different learning rates for the generator and discriminator networks to enhance overall network stability. Experimental results show that the model demonstrates superior performance in handling small targets and detailed features across different datasets. Although our method achieves significant improvements in this area, we recognize the limitations of its application exclusively to PCB inspection. We will continue our research to explore and achieve better performance.

Acknowledgements

Funding information: Basic Research Key Project of Liaoning Provincial Department of Education (LJ212410153035).

References

[1] X.C. Yan, J.M. Yang, K. Sohn, H. Lee, Attribute2Image: Conditional image generation from visual attributes, in: Proceedings of the 14th European Conference on Computer Vision, 2016.

[2] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, Improved techniques for training gans, in: Proc. Advances in neural information processing systems, 2016.

[3] C. Li, T. Xu, J. Zhu, B. Zhang, Triple generative adversarial nets, in: Proc. Advances in Neural Information Processing Systems 30, 2017.

[4] H. Fang, W. Deng, Y. Zhong, J. Hu, Triple-GAN: Progressive face aging with triple translation loss, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020.

[5] Z. Gan, L. Chen, W. Wang, Y. Pu, Y. Zhang, H. Liu, C. Li, L. Carin, Triangle generative adversarial networks, in: Proc. Advances in Neural Information Processing Systems 30, 2017.

[6] M. Mirza, S. Osindero, Conditional generative adversarial nets. <https://arxiv.org/abs/1411.1784>, 2014 (accessed 15.12.2022).

[7] A. Odena, C. Olah, J. Shlens, Conditional image synthesis with auxiliary classifier gans, in: Proc. International conference on machine learning, 2017.

[8] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, P. Abbeel, in: Proc. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. Advances in neural information processing systems, 2016.

[9] Y.-L. Lin, X.-Y. Dai, L. Li, X. Wu, F.-Y. Wang, The new frontier of AI research: generative adversarial networks, Acta Automatica Sinica 44(5)(2018) 775-792.

[10] A. Jabbar, X. Li, B. Omar, A survey on generative adversarial networks: Variants, applications, and training, ACM Computing Surveys 54(8)(2022) 1-49.

[11] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, A. Courville, Improved training of wasserstein gans, in: Proc. Advances in neural information processing systems, 2017.

[12] D. Berthelot, T. Schumm, L. Metz, Began: Boundary equilibrium generative adversarial networks. <https://arxiv.org/abs/1703.10717v1>, 2017 (accessed 15.03.2023).

[13] M. Sultana, A. Mahmood, T. Bouwmans, M.H. Khan, S.K. Jung, Moving objects segmentation using generative adversarial modeling, Neurocomputing 506(2022) 240-251.

[14] J. Redmon, You only look once: Unified, real-time object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[15] 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.

[16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.Y. Fu, A.C. Berg, SSD: Single Shot MultiBox Detector, in: Proceedings of Computer Vision–ECCV 2016: 14th European Conference, 2016.

[17] C.-Y. Wang, A. Bochkovskiy, H.-Y.M. Liao, YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023.

[18] D. Reis, J. Kupec, J. Hong, A. Daoudi, Real-Time Flying Object Detection with YOLOv8. <https://arxiv.org/abs/2305.09972v1>, 2023 (accessed 10.09.2023).

[19] A. Jahagirdar, N. Sathe, S. Thorat, S. Saxena, Fire detection in nano-satellite imagery using Mask R-CNN, International Journal of Signal and Imaging Systems Engineering 13(1)(2024) 19-26.

[20] F. Yang, L. Huang, X. Tan, Y. Yuan, FasterNet-SSD: A small object detection method based on SSD model, Signal, Image and Video Processing 18(1)(2024) 173-180.

[21] S. Wang, Z. Feng, Intelligent fault diagnosis of multi-sensor rolling bearings based on variational mode extraction and a lightweight deep neural network, International Journal of Signal and Imaging Systems Engineering 13(1)(2024) 27-40.

[22] Y.J. Cha, W. Choi, O. Buyukozturk, Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks, Computer-Aided Civil and Infrastructure Engineering 32(5)(2017) 361-378.