electronics

MDPI

Article

PreCaCycleGAN: Perceptual Capsule Cyclic Generative Adversarial Network for Industrial Defective Sample Augmentation

Jiaxing Yang [1,2], Ke Wang [1], Fengkai Luan [1,*], Yong Yin [1,2] and Hu Zhang [1]

1 School of Information Engineering, Wuhan University of Technology, Wuhan 430070, China; 276140@whut.edu.cn (J.Y.); 276028@whut.edu.cn (K.W.)
2 Chongqing Research Institute, Wuhan University of Technology, Chongqing 401151, China
* Correspondence: yimu@whut.edu.cn

Abstract: Machine vision is essential for intelligent industrial manufacturing driven by Industry 4.0, especially for surface defect detection of industrial products. However, this domain is facing sparse and imbalanced defect data and poor model generalization, affecting industrial efficiency and quality. We propose a perceptual capsule cycle generative adversarial network (PreCaCycleGAN) for industrial defect sample augmentation, generating realistic and diverse defect samples from defect-free real samples. PreCaCycleGAN enhances CycleGAN with a U-Net and DenseNet-based generator to improve defect feature propagation and reuse and adds a perceptual loss function and a capsule network to improve authenticity and semantic information of generated features, enabling richer and more realistic global and detailed features of defect samples. We experiment on ten datasets, splitting each dataset into training and testing sets to evaluate model generalization across datasets. We train three defect detection models (YOLOv5, SSD, and Faster-RCNN) with original data and augmented data from PreCaCycleGAN and other state-of-the-art methods, such as CycleGAN-TSS and Tree-CycleGAN, and validate them on different datasets. Results show that PreCaCycleGAN improves detection accuracy and rate and reduces the false detection rate of detection models compared to other methods on different datasets, demonstrating its robustness and generalization under various defect conditions.

Keywords: industrial surface defects; defect sample augmentation; perceptual capsule cycle generative adversarial network; defect detection models

check for updates

Citation: Yang, J.; Wang, K.; Luan, F.; Yin, Y.; Zhang, H. PreCaCycleGAN: Perceptual Capsule Cyclic Generative Adversarial Network for Industrial Defective Sample Augmentation. *Electronics* **2023**, *12*, 3475. https://doi.org/10.3390/electronics12163475

Academic Editor: Stefanos Kollias

Received: 11 July 2023
Revised: 8 August 2023
Accepted: 9 August 2023
Published: 16 August 2023

1. Introduction

Industry 4.0 is a new generation of industrial revolution with intelligent manufacturing as its core, which aims to achieve the integration of the physical world and the virtual network world through the deep application of information and communication technology and improve the sustainability and innovation of production [1]. Intelligent manufacturing uses information technology, artificial intelligence, the Internet of things, and other means to realize the digitalization, networking, automation, and intelligent management and control of the entire manufacturing process and flexibly adjust the production scale and product structure according to market demand, which can effectively improve resource utilization, reduce cost, ensure quality, enhance innovation ability, and promote sustainable development [2,3]. With industry 4.0 promoting the intelligent transformation of enterprises, machine vision has become widely used in various manufacturing processes. Machine vision uses computer vision and image processing methods to analyze and understand various images in industrial manufacturing [4]. It can perform functions such as monitoring, detection, identification, and measurement of production processes. Compared to manual vision, machine vision can effectively improve inspection speed and accuracy and reduce human resources and human errors [5]. Machine vision has a broad range of applications in diverse fields, such as automotive manufacturing, electronic components, food processing, textile printing and dyeing, and pharmaceutical and chemical industries [6]. One of the important applications of machine vision is the detection of surface defects in industrial products, where anomalies in material processing result in undesirable phenomena on the product surface, such as cracks, scratches, pits, and bubbles, which can impair the product appearance or function and may cause more serious consequences. Hence, it is essential to identify and reject defective products in a timely and accurate manner on the production line.

The difficulty of industrial defect detection lies in the fact that industrial products or parts differ in their surface materials, shapes, colors, and other characteristics, and the types, locations, sizes, and forms of defects also differ greatly [7]. Therefore, a general and flexible method that can adapt to different inspection scenarios and needs is required. Traditional industrial defect detection methods rely on techniques such as manual rules or template matching [8], which necessitate artificially set thresholds or templates and are not only time-consuming and labor-intensive but also difficult to adapt to different types and complexities of defects. In recent years, deep learning methods [9] have achieved breakthroughs in the field of computer vision and have demonstrated performance and potential to surpass traditional methods in the field of industrial defect detection. Deep learning methods can automatically learn high-level feature representations from large amounts of industrial defect data and use neural network models for classification, localization, or segmentation to accomplish defect detection tasks [10]. However, deep learning methods still encounter two main challenges and problems in the field of industrial defect detection [11]. The first challenge is the lack of industrial defect sample data [12]. Deep learning methods depend on a large amount of labeled data to train models, but in industrial scenarios, it is challenging to obtain sufficient and representative sample data due to the wide variety of products, complex types of defects, and strict production processes. In particular, there is a significant imbalance between normal and defective samples, which can lead to a bias toward normal samples during model training and thus affect the model's ability to identify defective samples. The second challenge is the poor generalization ability of the model [10]. The lack of defective datasets results in poor model training, which requires a model with strong generalization ability that can adapt to changes in different scenarios and maintain high detection accuracy and robustness. However, in practical applications, it is difficult to ensure good generalization ability of the model due to the quality and quantity of the dataset as well as the structure and parameters of the model, which leads to false detection or missed detection when the model is faced with new or unknown defects [13].

One of the key challenges for deep learning models of industrial defect detection is how to obtain diverse and high-quality industrial sample datasets. Adversarial Generative Networks (GAN) is an innovative and influential technique in deep learning, which consist of two neural networks: a generator and a discriminator. They use adversarial training to build unsupervised learning models that can learn and generate datasets with similar distribution characteristics to the training data without relying on prior information. The generator tries to produce samples that are indistinguishable from the real target samples, while the discriminator tries to distinguish between the generated samples and the real samples. The two networks compete with each other until they reach a Nash equilibrium. Currently, GAN and their derived models have gradually become a research hotspot in the field of data generation, mainly involving two research directions: models based on network architecture and loss function and models based on domain crossing. Models based on network architecture and loss function mainly improve the objective function and network structure of the original GAN to solve problems such as unstable training, mode collapse, etc., and improve the quality and diversity of the generated samples. The generator tries to minimize the discrimination error of the discriminator for its generated samples. The cGAN [14] proposed a GAN based on conditional probability distribution, which can input additional labels, texts, or image information as conditions to the generator and discriminator and improve the realism of the generated samples according to this information. The discriminator tries to maximize its ability to distinguish between real samples and generated samples. AC-GAN [15] proposed a GAN based on an auxiliary classifier, which adds an extra classifier in the discriminator to predict the category of the input sample, and combines the classification loss and discrimination loss to optimize the network. This can improve the quality and diversity of the generated samples and also use category information to control the generation process. The objective function of the original GAN is a minimax game, which has problems such as gradient vanishing, saddle point, KL divergence asymmetry, etc. Therefore, WGAN introduced Wasserstein distance as a measure between the real distribution and the generated distribution and gave a simple and effective algorithm to optimize this distance. WGAN [16] can avoid gradient vanishing and mode collapse and provide a meaningful indicator of training progress. Models based on domain crossing mainly use the relationship or transformation rules between different domains to achieve cross-domain generation tasks. In the original GAN design, the generator can only map from a random noise space to a data space and cannot achieve transformation between different data spaces. Pix2Pix [17] proposed a GAN based on conditional GAN and U-Net structure, which can achieve supervised transformation from one image domain to another image domain, such as from sketch to color image, from day to night, etc. StarGAN [18] proposed a GAN based on conditional GAN and CycleGAN structure, which can achieve unsupervised transformation between multiple image domains, such as changing facial expressions, hairstyles, gender, etc. These methods provide some pioneering suggestions for solving this kind of problem.

Although GAN has a wide range of applications in the sample augmentation field, they mainly focus on domains such as face attribute transformation and landscape color transformation. In the industrial defect detection field, due to difficulties such as lack of defect samples, low visibility of defects, irregular shape, unknown type, etc., existing GAN-based augmented samples are difficult to meet the task requirements of high accuracy and high speed at the same time. Therefore, designing a GAN model that can synthesize realistic and diverse defect samples with high fidelity and efficiency is a challenge in industrial defect detection. To address this challenge, we propose a perceptual capsule cyclic generative adversarial network (PreCaCycleGAN) for industrial defect sample augmentation, which aims to learn a more realistic distribution of industrial defect data. Our method leverages CycleGAN's framework of bi-directional mapping and cyclic consistency loss and enhances it with least-squares loss and perceptual loss function. Moreover, our method adopts an optimized generator structure with U-Net and DenseNet modules, and a capsule network with perspective invariance, to further improve the generator's ability to learn the features of industrial defect samples. The main contributions of our model are shown below:

(i) We design a generator model with U-Net network structure [19] and DenseNet [20] modules to enhance the feature propagation and feature reuse of defects. This can solve the gradient disappearance problem of deep networks and add perceptual loss functions to enhance the feature and semantic information of generated images;

(ii) We use cyclic consistency loss, identity mapping loss, and least squares loss to construct an adversarial training framework to achieve random changes in defect location and shape, ensure the consistency between the generated samples and the real samples in the non-defective region, improve the similarity between the generated samples and the real samples, and avoid the mode collapse and gradient vanishing or oscillation problems;

(iii) We design a discriminator model with PatchGAN [21] and capsule network [22] using dynamic routing protocols dual branches after the initial feature extraction, which can effectively extract and retain the detailed features of defective samples, identify the local and overall features of the samples, and improve the authenticity and diversity of industrial defect generation samples;

(iv) We compare our method with other generation algorithms and validate it in the actual industrial manufacturing defect detection model. We prove that our method has the optimal performance improvement for the actual industrial manufacturing defect detection model and can effectively increase the generalization ability of the defect detection model.

2. Related Work

Data augmentation is a common technique to enhance the performance and generalization ability of machine learning models by artificially creating new data to expand and enrich the training dataset. Sample augmentation is a specific form of data augmentation that is tailored to the characteristics and requirements of different domains or tasks. Data augmentation has been widely applied in computer vision, especially for tasks such as image classification, object detection, semantic segmentation, etc., where it can effectively address the issues of data insufficiency, dataset imbalance, and overfitting. However, in industrial defect detection, obtaining industrial defect samples is challenging due to the high yield rate of intelligent manufacturing, which leads to the lack of quantity and diversity of defect samples. Moreover, industrial defect samples require manual inspection and annotation by professionals, which is time-consuming and expensive. Furthermore, industrial defect samples have high complexity and diversity and are often sensitive and confidential, which restricts data sharing and communication and hinders the development of the industrial defect detection field. Therefore, designing suitable data enhancement techniques to overcome the data scarcity and imbalance problems in the industrial defects domain and to improve the robustness and accuracy of industrial defects detection models is an important and meaningful topic. We will review the current related research in data augmentation from three perspectives: Model-free image augmentation, Model-based image augmentation, and optimizing policy-based image augmentation, and analyze their advantages and challenges in industrial defect samples.

2.1. Model-Free Image Augmentation

Model-free Image Augmentation (MIA) is a data augmentation method that does not depend on any model training or optimization, and it augments the data by applying various geometric or color transformations to the original image, such as rotation, translation, scaling, cropping, flipping, brightness adjustment, contrast adjustment, etc. [23]. These transformations can be done in image space or frequency domain and can be randomly combined. However, these conventional transformation methods often only increase the data quantity but not the data diversity and may cause information loss or distortion. To address this issue, some researchers proposed methods such as CutMix [24], which mixes different images; Random Erasing [25], which replaces pixel values with random rectangles; Noise Injection [26], which adds random values from Gaussian distributions to an image; and Copy-Paste [27], which randomly pastes instance targets on background images. These methods can improve the data diversity and complexity by blending or erasing images, but they can also lose the details and boundary information of the images, which can affect the performance of the model for fine-grained target detection tasks.

MIA is a general and simple data augmentation method that can be applied to any image data and task, but there are few studies on algorithms specifically designed for industrial defect sample augmentation. Farady et al. [28] only proposed PreAugNet in 2023, which uses a Support Vector Machine (SVM) as a class boundary classifier to filter the samples generated by MIA and combine them with the original ones. The limitations of MIA for industrial defect sample augmentation are mainly divided into two aspects: on the one hand, it cannot customize the transformations for a specific type of defects, and it usually requires manual setting of the transformation types and parameters, which are hard to adapt to different tasks and datasets. On the other hand, it can only transform the original image in its spatial or frequency domain and cannot change the content or structure of the image, so the difference between the generated samples and the original samples is limited, and it cannot effectively extend the data distribution or cover a new feature space to generate new samples. This cannot cope with the industrial defect images with specific structures or constraints that are generated from complex and dynamic industrial defect scenarios, and excessive transformations may destroy the semantic information of the image and thus compromise the quality and authenticity of the generated industrial defect samples.

2.2. Optimizing Policy-Based Image Augmentation

Optimizing Policy-based Image Augmentation (OPIA) is an approach that uses an optimization algorithm to search for the optimal data augmentation policy. OPIA is essentially a sequence of MIA operations and their parameters, such as rotating 15 degrees + crop 0.8 + brightness adjustment 0.2, etc. OPIA can automatically find the best data augmentation strategy for different datasets and tasks and can significantly improve the model performance on a test set. Cubuk et al. [29] proposed AutoAugment, the first OPIA method, which uses a reinforcement learning-based controller to select the optimal data augmentation strategy, but it is very slow and computationally intensive. Cubuk et al. [30] then proposed RandAugment based on the data augmentation strategy of the Neural Network Architecture Search (NAS) method [31], which reduces the search space, makes the search results more general and stable, and can adapt to models and datasets of different sizes and complexities. Lim et al. [32] further improved AutoAugment by proposing Fast AutoAugment, which uses Bayesian optimization and density matching to speed up the search process and is three orders of magnitude faster than AutoAugment in search time while achieving similar or better performance. Ho et al. [33] proposed Population-Based Augmentation, which optimizes both the target network and the data augmentation strategy, and PBA is four orders of magnitude faster than AutoAugment in search time while achieving similar or better performance. Zhang et al. [34] proposed Adversarial AutoAugment based on Adversarial Production Networks, which uses adversarial loss and reinforcement learning to optimize the data augmentation strategy, and Adversarial AutoAugment is 12 times faster than AutoAugment in search time while achieving the best performance on multiple datasets. However, OPIA still depends on MIA as a transformation operation and thus suffers from the same problems and limitations faced by model-free image augmentation techniques. For industrial defect detection, no studies have been found using optimization strategy-based image augmentation techniques to improve model performance. This may be due to the lack of sufficiently large and high-quality training data and feedback signals in industrial defect detection, which makes it difficult for optimization strategy-based image augmentation techniques to effectively learn data augmentation strategies or parameters.

2.3. Model-Based Image Augmentation

Model-based Image Augmentation (MBIA) is an approach that leverages deep learning models to synthesize new data samples. With the advancement of deep learning, traditional data augmentation methods are gradually replaced by data augmentation algorithms based on deep learning frameworks. Deep learning models can learn latent feature distributions from raw data and can generate new data samples from random noise or conditional inputs. MBIA can effectively increase the size and diversity of datasets and can produce high-quality and high-fidelity data samples. Kuo et al. [35] proposed FeatMatch based on Convolutional Neural Networks (CNNs) [36], which replaces simple transformations in image space with complex transformations generated in feature space to achieve data augmentation effects in feature space, thus enhancing the data diversity and consistency. However, the lack of interpretability of vector data in feature space leads to difficult and time-consuming training. Therefore, Wong et al. [37] changed the perspective of data augmentation to data space and found that data augmentation in data space is superior to data augmentation in feature space. However, both data augmentation methods in feature space and data space do not sufficiently learn the true distribution of the sample data, which makes the data augmentation methods based on Adversarial Generative Networks (GANs) [38] start to attract attention and research.

GANs is a deep learning framework that consists of generative and discriminative models that compete with each other. GANs can learn the underlying data distribution from raw samples and generate novel samples with diverse attributes such as types, positions, sizes, and shapes. The generation and discrimination processes are driven by a zero-sum game that ensures the progressive convergence between the generated and authentic data distributions. However, the GAN training process faces many challenges due to its non-convex and non-cooperative nature. Mode collapse, gradient vanishing, and oscillatory disturbances are common problems that affect the quality and diversity preservation in generated samples. Various GAN variants such as WGAN [16], LS-GAN [39], and f-GAN [40] have introduced different loss functions and distance metrics to improve the similarity between generated and real samples. Likewise, models such as cGAN [14], AC-GAN [15], and InfoGAN [41] have modified the architectures of generators and discriminators to increase the expressiveness and diversity of generative models. However, due to the complex and variable features of industrial defects, relying only on GANs to generate new industrial defect samples from random noise might lead to significant differences or biases compared to real samples. The generated samples might lack plausibility or credibility, which limits the application of GANs in industrial defect sample generation.

To endow GANs with more control mechanisms for sample generation, models such as Pix2Pix [17] and CycleGAN [21] have used translation between different images to impose constraints on generated samples, ensuring their closer approximation to real images. Based on this idea, some researchers have explored industrial defect sample generation. Qin et al. [42] proposed Tree-CycleGAN, a cyclic generative adversarial network based on a symmetric tree structure. This method uses a tree-structured generator with maximal diversity loss to enable one-to-many generation mappings. Using a tree-structured reconstructor and dual discriminators, Tree-CycleGAN can generate multiple target domain samples from a single source domain sample while preserving differences and cyclic consistency across different branches. This method effectively alleviates the problem of industrial defect sample insufficiency.

Similarly, Song et al. [43] introduced CycleGAN-TSS, a Texture Self-Supervised CycleGAN that leverages texture information as a self-supervisory signal to guide the generator in acquiring enhanced shadow features. Compared to traditional CycleGAN, CycleGAN-TSS can produce more realistic shadow images, thereby improving road crack detection performance. Niu et al. [44] proposed a method that combines CycleGAN with a Defect Attention Module (DAM). This adaptive method adjusts the weights of defect regions and integrates structural similarity (SSM) into the original L1 loss to formulate the Defect Cycle Consistency Loss (DCL). By using grayscale and structural features, this method enhances the simulation of internal defect structures. Notably, unlike other GAN-based methods, this method yields clearer and more authentic defect images, thereby enhancing defect detection accuracy. In a different contribution, SHAO et al. [45] introduced DuCaGAN, a Dual Capsule Generative Adversarial Network based on CycleGAN. DuCaGAN uses the Dual Capsule Network (DCN) [22] to generate diversified and high-fidelity industrial defect samples, which can be used for practical industrial data augmentation.

All these methods address the problem of industrial defect sample augmentation to some extent, but in real industrial manufacturing applications, they often suffer from low quality, low diversity, and low fidelity and do not adequately reflect the data distribution of the real industrial defect samples, which affects the detection accuracy and generalization of the deep learning-based defect detection model. Therefore, MBIA needs to design appropriate network structures and loss functions to adapt to different data characteristics and task requirements and balance the relationship between quality and diversity of generated samples while avoiding training difficulties such as mode collapse and gradient vanishing.

3. The Proposed Model

3.1. Overall Structure

To address the challenge of small sample sizes in industrial defect detection and to address the shortcomings of the current model-based image augmentation methods, we present a model PreCaCycleGAN that leverages defect-free samples to synthesize defective samples based on CycleGAN, as illustrated in Figure 1.

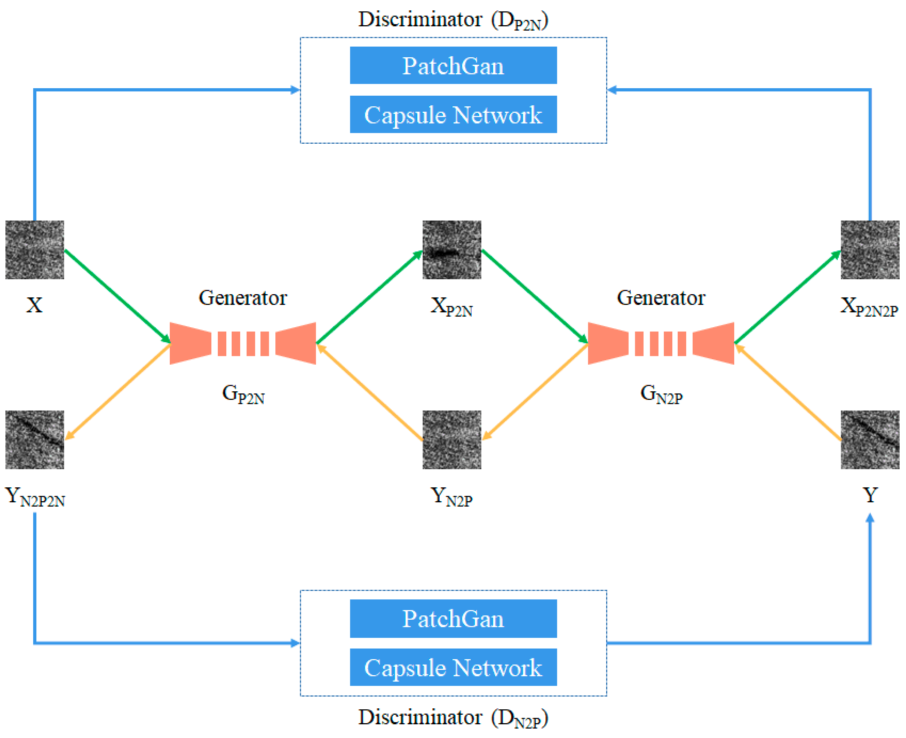

Figure 1. Schematic diagram of PreCaCycleGAN structure.

The framework includes two generators D_{P2N} (Positive Sample to Negative sample) and D_{N2P} (Negative sample to Positive Sample), and two discriminators D_{P2N} and D_{N2P}. The generators are optimized based on the U-Net network structure, and the perceptual loss function is incorporated as a constraint to enhance the feature and semantic quality of the generated images. In the discriminator, we employ capsule networks to learn more refined global spatial features based on PatchGAN. Furthermore, we replace the Sigmoid cross-entropy loss function with least squares to overcome the gradient vanishing problem during training and prevent mode collapse and training instability.

3.2. Generator Structure

The generator G architecture is illustrated in Figure 2. Initially, the input image is subjected to channel expansion by a convolution layer employing convolution operation, and the convolution kernel of this layer possesses a size of 3×3 and a stride of 1. Subsequently, the feature map is aggregated and reconstructed by four times downsampling and four times upsampling, and ultimately the defective samples are synthesized by the activation layer. To augment the local detail feature extraction of the defective samples and enhance the network training efficiency and accuracy, we incorporate the summation operation with the antecedent layer prior to transmitting to the subsequent layer in the first three layers of downsampling and amalgamate with the residual module [46] to further ascertain the feature integrity of the samples in the downsampling process. We adopt the DenseNet Block [20] in the converter layer in lieu of the ResNet structure to considerably diminish the parameter and computation overheads. In the upsampling process, we exploit the upsampling module to accomplish the stitching of the downsampled feature maps of the corresponding scales through skip connections and fuse the features by the residual module based on the nearest interpolation upsampling before conveying them to the next layer.

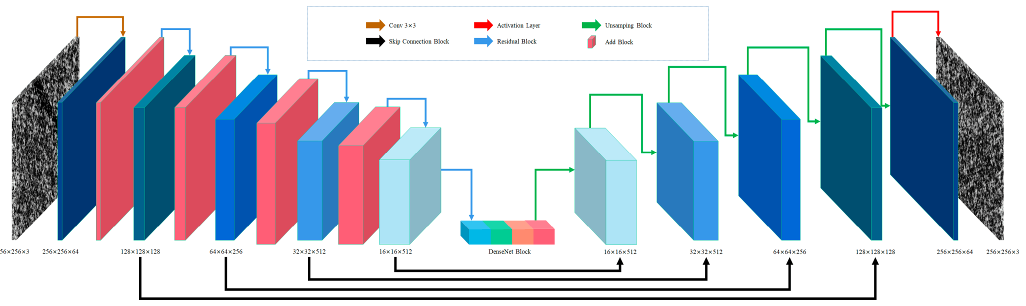

Figure 2. Schematic diagram of PreCaCycleGAN generator structure.

3.3. Discriminator Structure

We proposed a discriminator D with a PatchGAN and a capsule network with two branches [22] to optimize the discriminative output, as illustrated in Figure 3. The input image undergoes three feature extraction layers and then bifurcates into two branches for the output. The first branch employs the original PatchGAN discriminator structure to assess the local authenticity of the image. The second branch utilizes the capsule network to achieve sample discrimination and evaluate the global consistency of the image.

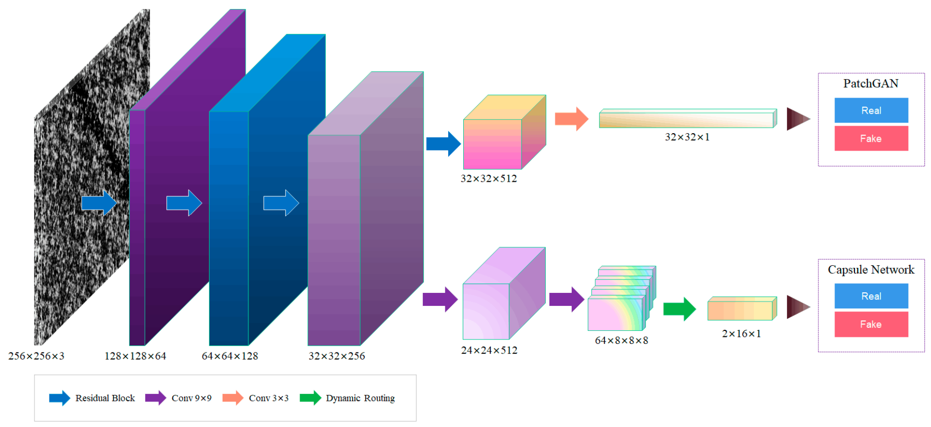

Figure 3. Schematic of PreCaCycleGAN discriminator structure.

To better preserve the spatial information of industrial defect samples, we employed vector encoding of the primary capsule layer and the digit capsule layer to represent the probability of feature existence and spatial information in the capsule network. This enhances the realism and diversity of the generated defect samples, as well as their interpretability and controllability. We also employed a dynamic routing mechanism between two consecutive capsule layers to iteratively learn and predict the features of the lower layer and achieve an adaptive feature combination. The dynamic routing relationship between capsule i in layer l and capsule j in layer (l + 1) is depicted in Equation (1).

$$\hat{u}_{j|i} = W_{ij} \cdot u_i \tag{1}$$

where the output of capsule i is u_i, the weight matrix between capsule i and capsule j is W_{ij}, and the prediction vector from capsule i to capsule j is $\hat{u}_{j|i}$. The discriminator only needs to output two types of data, true or false, and j takes the value {0,1}. i is then determined by the total number of master capsules and takes the value $\{1 \leq i \leq 4096 \mid i \in \mathrm{N}\}$.

After the prediction vectors are input to the dynamic routing incentive mechanism, for each prediction vector, a routing weight needs to be defined, which is the log prior probability between capsule i and capsule j. The coupling coefficient is obtained using softmax, as shown in Equation (2).

$$c_{ij} = \frac{exp(b_{ij})}{\Sigma_k exp(b_{ij})} \tag{2}$$

where c_{ij} is the coupling coefficient of the prediction vector and b_{ij} is the routing weight. Since there is no initial routing preference, the initial routing weight of each capsule is the same, and the sum of the coupling coefficients of all prediction vectors is 1, so the initial value of b_{ij} is set to 0. So the output sum s_j of capsule j in the $(l + 1)$th layer is shown in Equation (3).

$$s_j = \Sigma_i c_{ij} \cdot \hat{u}_{j|i} \tag{3}$$

where s_j is the output vector of layer j, representing the sum probability of all weighted prediction vectors in this layer, and it is necessary to use the squash function for s_j to ensure that the probability takes values between [0, 1], as shown in Equation (4).

$$v_j = \frac{\|s_j\|^2}{1 + \|s_j\|^2} \cdot \frac{s_j}{\|s_j\|} \tag{4}$$

where, v_j is the predicted probability output of layer j after compression, and in the calculation, in order to prevent the denominator from being 0, the denominator is preprocessed by adding ε, where ε is taken as 10^{-8}, as shown in Equation (5).

$$\|s\| \approx \sqrt{\sum_i s_j^2 + \varepsilon} \tag{5}$$

The routing iteration protocol is represented by the dot product of the output vector and the prediction vector, and the larger the dot product represents the smaller the pinch angle, which proves that the consistency of the output vector and the prediction vector is better, and the protocol a_{ij} is shown in Equation (6).

$$a_{ij} = v_j \cdot \hat{u}_{j|i} \tag{6}$$

The dynamic routing incentive mechanism is a cyclic structure, so the routing weights b_{ij} need to be updated before the next cycle, and the formula is shown in Equation (7).

$$b_{ij} \leftarrow b_{ij} + a_{ij} \tag{7}$$

We used a_{ij} to measure the consistency between the output vector and the prediction vector. The higher the a_{ij}, the higher the coupling coefficient c_{ij} is updated, and the higher the probability that capsule i is assigned to capsule j. This means that capsule j is more likely to be activated and to represent the existence of an entity. The dynamic routing mechanism replaces the scalar output feature detector of the convolutional neural network with a vector output, replaces the max pooling layer with a routing protocol mechanism, and optimizes the discriminative output. The number of routing iterations is denoted by r; here, r = 3. The algorithm of the dynamic routing mechanism is shown in Algorithm 1.

The dynamic routing mechanism processes each capsule in the primary capsule layer and then iteratively learns and predicts the features of the next layer. The activation capsule vector for each layer is found, and the output value of the primary capsule layer is obtained by continuous iterative updates.

Algorithm 1 Dynamic Routing Algorithm

```
1.  Input: vector neuron prediction capsule $\hat{u}_{j|i}$, iteration number r
2.  Output: vector neuron output vector $v_j$
3.      Routing weights initialization assignment: $b_{ij} \leftarrow 0$
4.      for k in r do
5.        Coupling coefficients corresponding to all prediction capsules: $c_{ij} \leftarrow softmax(b_i)$
6.        Weighted summation of all prediction capsules: $s_j \leftarrow \Sigma_i c_{ij} \cdot \hat{u}_{j|i}$
7.        Normalized compression of pairs: $v_i \leftarrow squash\left(s_j\right)$
8.        Update routing weights: $b_{ij} \leftarrow b_{ij} + a_{ij}$
9.      end for
```

3.4. Loss Function

The CycleGAN model generation involves two types of loss functions: the adversarial loss function and the reconstruction loss function. The adversarial loss function aims to minimize the discrepancy between the data distributions of the generated image and the target domain, thus producing more realistic images. The reconstruction loss function ensures that the mapping relation between the source and target domains is consistent and aligned. To enhance the quality of the industrial defect images, we incorporate the perceptual loss function and the capsule loss function based on the U-Net structure and the capsule network structure. We also add the identity mapping loss to better capture the industrial defect features. The overall loss function is given by Equation (8).

$$\begin{aligned} L_{PreCaCycleGAN} = \; & L_{GAN}(G_{P2N}, D_{P2N}, X, Y) + L_{GAN}(G_{N2P}, D_{N2P}, Y, X) \\ & + \lambda_1 L_{cycle}(G_{P2N}, G_{N2P}, X, Y) + \lambda_2 L_{identity}(G_{P2N}, G_{N2P}, X, Y) \\ & + \lambda_3 L_{perceptual}(G_{P2N}, G_{N2P}, X, Y) \end{aligned} \tag{8}$$

Standard generative adversarial networks adopt a binary, zero-sum game, which poses a very large very small game problem [38]. In this game, the generator and the discriminator compete to reach the final Nash equilibrium, as shown in Equation (9). The CycleGAN model employs the Sigmoid cross-entropy loss function as the adversarial loss function, which is suitable for logical classification problems. However, this loss function can cause gradient vanishing problems in the training of generative adversarial networks, affecting model convergence and optimization. To address this issue, we use the least squares method as the adversarial loss function and combine it with the edge loss of the capsule network. This improves the stability and convergence of the training and ensures the authenticity and diversity of the generated samples. The model training loop consists of defective industrial samples and non-defective industrial samples that are mutually generated with the same two parts of the loss function. As an example, we introduce the following formulas to generate defective samples (y) from non-defective samples (x) by using the generator G_{P2N} and the discriminator D_{P2N}, and we show the adversarial loss function in Equation (10).

$$G*, D* = arg \min_{\mathrm{G}} \max_{\mathrm{D}} \mathrm{L}(G_{P2N}, D_{P2N}, G_{N2P}, D_{N2P}, X, Y) \tag{9}$$

$$\begin{aligned} & L_{GAN}(G_{P2N}, D_{P2N-1}, D_{P2N-2}, X, Y) \\ & \qquad = E_{x\sim P_{data}(x)}\left[(D_{P2N-1}(G(x)) - 1)^2\right] \\ & \qquad + E_{y\sim P_{data}(y)}\left[(D_{P2N-1}(y) - 1)^2\right] + E_{x\sim P_{data}(x)}\left[(D_{P2N-1}(G(x)))^2\right] \\ & \qquad + \lambda E_{y\sim P_{data}(y)}[-L_M D_{P2N-2}(y), T = 1] \\ & \qquad + \lambda E_{x\sim P_{data}(x)}[-(D_{P2N-2}(G(x))), T = 0] \end{aligned} \tag{10}$$

where D_{P2N-1} is the discriminator branch that uses the PatchGAN structure and D_{P2N-2} is the discriminator branch that uses the capsule network. The first part on the right-hand side of the equal sign is the loss function corresponding to the generator, and its optimization objective is to make the value $D_{P2N-1}(G(x))$ of the discriminator that discriminates the generated image approach 1. The second and third parts are the loss functions corresponding to the discriminator, and their optimization objective is to make the value of the discriminator that discriminates the real image $D_{P2N-1}(G(x))$ approach 1, and the value of the discriminator that discriminates the generated image $D_{P2N-1}(G(x))$ approach 0. The fourth and fifth parts are the edge loss functions, and λ is a hyperparameter that indicates the relative importance of the edge loss in the improved adversarial loss. The capsule discriminator only needs to determine whether the input image is a real image or a generated fake image, which is defined in Equation (11).

$$L_M(k,v) = T_K \max\left(0, m^+ - \|v_k\|\right)^2 + \lambda(1 - T_K)\max\left(0, \|v_k\| - m^-\right)^2 \tag{11}$$

where v_k (see Equation (4)) is the output vector of the discriminator layer of the capsule network. k = 0 for real data, or k = 1 for generated false data. $T_K = 1$ if it is desired for the discriminator or generator to determine that this is true data at this point, or $T_K = 0$ if it is desired for the discriminator to determine that this is generated false data at this point. m^+ and m^- are the baselines for determining whether the input image is true or false. If the mode of the vector is larger than m^+, 0 is returned; if the mode of the vector is smaller than m^+, the square of the difference between the two is returned; if the mode of the vector is smaller than m^-, 0 is returned; if the mode of the vector is larger than m^-, the square of the difference between the two is returned.

In order to avoid the pattern collapse problem in the process of fighting against the loss function to reach Nash equilibrium, we add the perceptual loss function to enhance the feature and semantic information of the generated image during the generator training to make the generated image more realistic and clear. The formula is shown in Equation (12), where ϕ is the pre-trained VGG19 feature extractor.

$$L_{perceptual}(G_{P2N}, G_{N2P}, X, Y) = \|\phi(x) - \phi(G_{P2N}(G_{N2P}(x)))\|_2^2 + \|\phi(y) - \phi(G_{N2P}(G_{P2N}(y)))\|_2^2 \tag{12}$$

Identity loss and loop loss are used to ensure that the generated industrial defect images are consistent with the input industrial defect-free images in terms of content and structure, and the constraint generator generates industrial defect samples on the same background as the input defect-free samples, and the formulas are shown in Equations (13) and (14).

$$L_{cycle}(G_{P2N}, G_{N2P}, X, Y) = E_{x\sim P_{data}(x)}\|G_{P2N}(G_{N2P}(x)) - x\|_1 + E_{y\sim P_{data}(y)}\|G_{N2P}(G_{P2N}(y)) - y\|_1 \tag{13}$$

$$L_{identity}(G_{P2N}, G_{N2P}, X, Y) = E_{x\sim P_{data}(x)}\|G_{N2P}(x) - x\|_1 + E_{y\sim P_{data}(y)}\|G_{P2N}(y) - y\|_1 \tag{14}$$

The training process of the PreCaCycleGAN model is shown in Algorithm 2. Unlike the standard CycleGAN model, in order to ensure the convergence and accuracy of the model, we get the optimal training steps through experiments. In steps 3 to 6, we update the optimized generator G_{P2N}, discriminator D_{P2N}, generator G_{N2P}, discriminator D_{N2P}, to obtain the final industrial defect sample generation model.

Algorithm 2 PreCaCycleGAN Algorithm

```
Optimization Objective: generator $G_{P2N}$, discriminator $D_{P2N}$, generator $G_{N2P}$, discriminator $D_{N2P}$
1.  Initialize all network and hyperparameters
2.    for number of epochs do
          Draw a minibatch of samples $\{x^{(1)}, \ldots\ldots, x^{(m)}\}$ from domain X
          Draw a minibatch of samples $\{y^{(1)}, \ldots\ldots, y^{(m)}\}$ from domain Y
3.        Calculate generator $G_{P2N}$ loss and update all parameters of the generator $G_{P2N}$ by minimizing the generator loss
4.        Calculate generator $D_{P2N}$ loss and update all parameters of the generator $D_{P2N}$ by minimizing the generator loss
5.        Calculate generator $G_{N2P}$ loss and update all parameters of the generator $G_{P2N}$ by minimizing the generator loss
6.        Calculate generator $D_{N2P}$ loss and update all parameters of the generator $D_{N2P}$ by minimizing the generator loss
7.        end for
8.    end for
```

4. Validation Experiments

To assess the effectiveness of PreCaCycleGAN-generated defect samples in enhancing the generalization performance of real industrial defect detection models, we employed the DAGM 2007 dataset as an experimental platform. DAGM 2007 [47] is a publicly accessible dataset of texture surface images with various types of defects, which simulates the real-world defect detection problem with high complexity and diversity. We compared the defect samples produced by PreCaCycleGAN with those produced by Tree-CycleGAN [42] and CycleGAN-TSS [43] and applied them to three state-of-the-art defect detection models that are widely adopted in practice, namely YOLOv5 [48], SSD [49], and Faster-RCNN [50]. We evaluated the impact of PreCaCycleGAN-generated defect samples on the generalization performance of defect detection models by measuring the mAP, false detection rate, and other metrics of the three generative models on different defect detection models and datasets. All experiments were conducted on a single NVIDIA GeForce GTX 3060 GPU.

The training process lasted for 150 iterations with a batch size of 1. The learning rate was initially set to 0.0002 and was linearly decayed starting from the 100th iteration. The hyperparameters in the loss function were empirically determined. We set λ_1 to 10, λ_2 to 0.5, λ_3 to 0.02, m^+ in the capsule network to 0.9, m^- to 0.1, and λ to 0.5, respectively, and used the Adam optimizer with default parameters for gradient computation. The comparison plots of defects generated by the three generative models are shown in Figure 4.

Visually, compared to the other two models, the PreCaCycleGAN model with DenseNet incorporated into the U-Net network exhibits more diverse defect sample generation in four datasets: (b), (h), (i), (j). Moreover, in three datasets: (a), (e), (f), and (g), our model with a two-branch discriminator demonstrates a more refined feature representation. However, in the remaining dataset, our defect generation performance is not clearly superior to that of the other models.

Although our model appears to improve defect generation diversity and feature quality compared to the existing models from visual inspection, the generated defect samples need to be tested on actual industrial inspection models to verify their effectiveness. Therefore, we need further quantitative data to support the current superiority of our model.

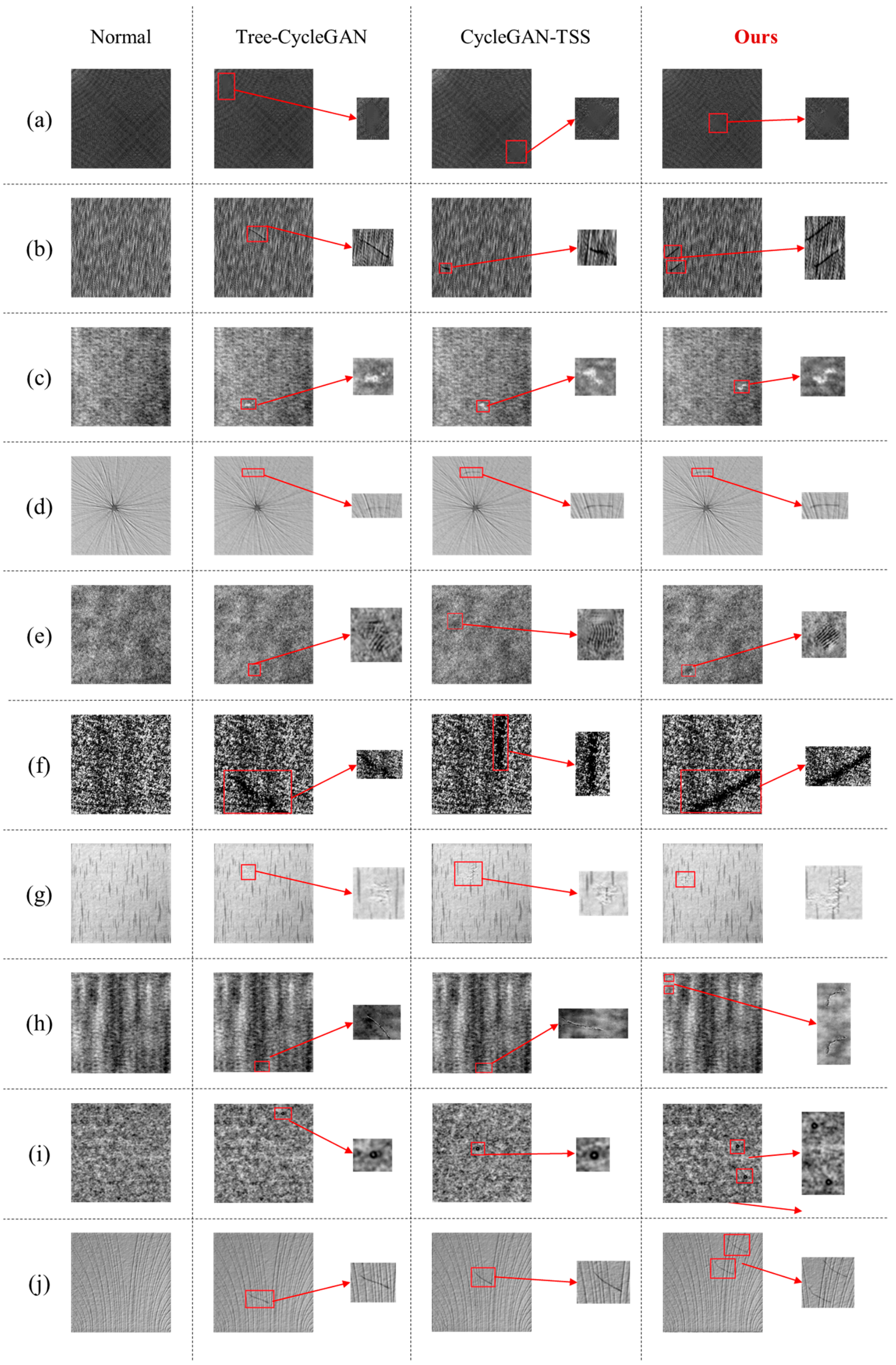

Figure 4. (**a**–**e**) represent the datasets of Class 1–Class 5, respectively. Comparison of generation defects of PreCaCycleGAN with different generative models on different datasets. Red boxes identify the defects in the images and are shown zoomed in adjacent positions. In (**a**,**b**,**e**), it can be clearly found that PreCaCycleGAN is able to generate more realistic and diverse defects, while the other models suffer from blurring and distortion. (**f**–**j**) represent Class 5–Class 10 datasets, respectively. In (**h**–**j**), it can be clearly found that PreCaCycleGAN is able to generate a wider variety and a larger number of defects, while the other models can only generate a single defect.

4.1. Detection Model Training Validation

To validate the effectiveness of our model-generated defect samples in real industrial manufacturing, we selected three models YOLOv5 [48], SSD [49], and Faster-RCNN [50], which are currently widely used in industrial inspection, for generating images for generalization enhancement of the detection model. In the training set, we designed three types of training sets De-Train A, De-Train B, and De-Train C, for different datasets, where De-Train A consists of 700 defect-free samples and 100 real defect samples, De-Train B consists of 700 defect-free real samples and 100 synthetic defect samples generated by different models, De-Train C consists of 700 defect-free samples, 50 real defect samples and 50 synthetic defect samples generated by different models. Tables 1–3 show the corresponding detection accuracies for the three training sets De-Train A, De-Train B, and De-Train C.

Table 1. Training detection accuracy of De-Train A.

Dataset	YOLOv5-mAP@0.5	SSD-mAP@0.5	Faster-RCNN-mAP@0.5
Class1	**0.667**	0.545	0.634
Class2	0.784	0.735	**0.791**
Class3	0.723	0.661	**0.754**
Class4	0.542	**0.596**	0.552
Class5	**0.654**	0.614	0.629
Class6	**0.735**	0.663	0.671
Class7	**0.720**	0.597	0.714
Class8	**0.584**	0.493	0.563
Class9	0.702	0.695	**0.712**
Class10	0.715	**0.733**	0.701

Table 2. Training detection accuracy of De-Train B.

Model	YOLOv5-mAP@0.5	SSD-mAP@0.5	Faster-RCNN-mAP@0.5
Tree-CycleGAN [42]$_{\text{Class1}}$	0.670	0.552	0.646
CycleGAN-TSS [43]$_{\text{Class1}}$	0.692	0.562	0.654
PreCaCycleGAN$_{\text{Class1}}$	**0.723**	**0.596**	**0.682**
Tree-CycleGAN [42]$_{\text{Class2}}$	0.811	0.781	0.828
CycleGAN-TSS [43]$_{\text{Class2}}$	0.794	0.762	0.817
PreCaCycleGAN$_{\text{Class2}}$	**0.834**	**0.798**	**0.842**
Tree-CycleGAN [42]$_{\text{Class3}}$	0.732	0.711	0.795
CycleGAN-TSS [43]$_{\text{Class3}}$	0.751	**0.729**	0.806
PreCaCycleGAN$_{\text{Class3}}$	**0.763**	0.726	**0.811**
Tree-CycleGAN [42]$_{\text{Class4}}$	0.560	0.622	0.578
CycleGAN-TSS [43]$_{\text{Class4}}$	**0.584**	**0.652**	0.596
PreCaCycleGAN$_{\text{Class4}}$	0.576	0.645	**0.602**
Tree-CycleGAN [42]$_{\text{Class5}}$	**0.679**	0.639	**0.654**
CycleGAN-TSS [43]$_{\text{Class5}}$	0.664	0.618	0.635
PreCaCycleGAN$_{\text{Class5}}$	0.674	**0.643**	0.651
Tree-CycleGAN [42]$_{\text{Class6}}$	0.746	0.687	0.695
CycleGAN-TSS [43]$_{\text{Class6}}$	0.763	0.709	0.727
PreCaCycleGAN$_{\text{Class6}}$	**0.791**	**0.728**	**0.745**
Tree-CycleGAN [42]$_{\text{Class7}}$	0.731	0.622	0.739
CycleGAN-TSS [43]$_{\text{Class7}}$	0.729	0.625	0.736
PreCaCycleGAN$_{\text{Class7}}$	**0.761**	**0.667**	**0.763**
Tree-CycleGAN [42]$_{\text{Class8}}$	0.608	0.539	0.593
CycleGAN-TSS [43]$_{\text{Class8}}$	0.592	0.533	0.596
PreCaCycleGAN$_{\text{Class8}}$	**0.614**	**0.548**	**0.602**
Tree-CycleGAN [42]$_{\text{Class9}}$	0.729	0.736	0.754
CycleGAN-TSS [43]$_{\text{Class9}}$	0.731	0.733	0.751
PreCaCycleGAN$_{\text{Class9}}$	**0.736**	**0.745**	**0.763**
Tree-CycleGAN [42]$_{\text{Class10}}$	0.747	0.765	0.738
CycleGAN-TSS [43]$_{\text{Class10}}$	0.731	0.775	0.745
PreCaCycleGAN$_{\text{Class10}}$	**0.756**	**0.781**	**0.749**

Table 3. Training detection accuracy of De-Train C.

Model	YOLOv5-mAP@0.5	SSD-mAP@0.5	Faster-RCNN-mAP@0.5
Tree-CycleGAN [42]$_{\text{Class1}}$	0.712	0.586	0.679
CycleGAN-TSS [43]$_{\text{Class1}}$	0.763	0.603	0.684
PreCaCycleGAN$_{\text{Class1}}$	**0.796**	**0.634**	**0.735**
Tree-CycleGAN [42]$_{\text{Class2}}$	0.856	0.826	0.868
CycleGAN-TSS [43]$_{\text{Class2}}$	0.839	0.817	0.862
PreCaCycleGAN$_{\text{Class2}}$	**0.881**	**0.832**	**0.887**
Tree-CycleGAN [42]$_{\text{Class3}}$	0.777	0.756	0.840
CycleGAN-TSS [43]$_{\text{Class3}}$	0.796	**0.774**	0.851
PreCaCycleGAN$_{\text{Class3}}$	**0.808**	0.771	**0.856**
Tree-CycleGAN [42]$_{\text{Class4}}$	0.605	0.667	0.623
CycleGAN-TSS [43]$_{\text{Class4}}$	**0.629**	0.687	0.641
PreCaCycleGAN$_{\text{Class4}}$	0.621	**0.690**	**0.647**
Tree-CycleGAN [42]$_{\text{Class5}}$	**0.724**	0.684	**0.699**
CycleGAN-TSS [43]$_{\text{Class5}}$	0.709	0.663	0.678
PreCaCycleGAN$_{\text{Class5}}$	0.719	**0.688**	0.694
Tree-CycleGAN [42]$_{\text{Class6}}$	0.791	0.732	0.740
CycleGAN-TSS [43]$_{\text{Class6}}$	0.808	0.757	0.772
PreCaCycleGAN$_{\text{Class6}}$	**0.835**	**0.776**	**0.790**
Tree-CycleGAN [42]$_{\text{Class7}}$	0.776	0.667	0.784
CycleGAN-TSS [43]$_{\text{Class7}}$	0.774	0.670	0.781
PreCaCycleGAN$_{\text{Class7}}$	**0.805**	**0.712**	**0.809**
Tree-CycleGAN [42]$_{\text{Class8}}$	0.653	0.584	0.638
CycleGAN-TSS [43]$_{\text{Class8}}$	0.637	0.578	0.641
PreCaCycleGAN$_{\text{Class8}}$	**0.659**	**0.592**	**0.646**
Tree-CycleGAN [42]$_{\text{Class9}}$	0.774	0.781	0.799
CycleGAN-TSS [43]$_{\text{Class9}}$	0.776	0.779	0.796
PreCaCycleGAN$_{\text{Class9}}$	**0.783**	**0.788**	**0.807**
Tree-CycleGAN [42]$_{\text{Class10}}$	0.772	0.790	0.763
CycleGAN-TSS [43]$_{\text{Class10}}$	0.756	0.792	0.776
PreCaCycleGAN$_{\text{Class10}}$	**0.781**	**0.806**	**0.780**

Out of the 60 results from De-Train B training and De-Train C training, PreCaCycleGAN achieved 51 top scores, and the results in the items that did not achieve top scores were very close to the top scores. In general, the detection accuracy was improved by 3–5% using De-Train B to train the dataset than using De-Train A to train the dataset, while the detection accuracy was improved by 8–10% using De-Train C to train the dataset than using De-Train A to train the dataset, which proves that the defect samples generated by different models all have a significant impact on the generalization and accuracy of the detection model. The improvement is evident, especially when the mixture of generated defect samples and real defect samples is used to train the detection model, which is consistent with the current actual industrial manufacturing situation.

We further compared the defect samples generated by PreCaCycleGAN with those generated by Tree-CycleGAN and CycleGAN-TSS and found that PreCaCycleGAN-generated defect samples exhibited better detail features for detection model learning in different datasets. Taking the YOLOv5 detection model as an example, we observed that PreCaCycleGAN-generated defect samples improved the detection accuracy by about 4% in (a), (b), (f), (g), and by 1–2% in the remaining datasets, compared to the other two generative models. This proves that our model can generate images with more detailed defect features and defect diversity. The same trend was observed in both SSD and Faster-RCNN detection models, demonstrating that our model-generated images can be practically applied to industrial defect detection models and show good generalization.

4.2. Detection Model Test Validation

To further verify the generalization improvement of the generated images to the detection model, test sets are constructed to show the application performance of the detection model in multiple dimensions and to demonstrate the practicality of our model to generate defective samples. In the validation set, we also set three types of test sets De-Test A, De-Test B, and De-Test C, for different datasets, where De-Test A is composed of 120 defect-free samples and 60 real defect samples, De-Test B is composed of 120 defect-free real samples and 60 fake defect samples generated by different models. De-Test C is composed of 120 defect-free samples, 30 real defective samples, and 30 fake defective samples generated by different training sets and training models correspondingly. During the experiments, we used the three most important metrics in real industrial manufacturing, data detection accuracy (DDA), defect detection rate (DDR), and false detection rate (FDR), to measure the detection accuracy [51]. Among them, the data detection accuracy is the percentage of the sum of correctly detected defect data and defect-free data in the total data volume, the defect detection rate is the percentage of correctly detected defects in the total defect data, and the false detection rate is the percentage of incorrectly detected defect-free samples as defective samples among all samples detected as defective, as shown in Equations (15)–(17).

$$DDA = \frac{TP + TN}{TP + TN + FP + FN} \tag{15}$$

$$DDR = \frac{TP}{TP + FN} \tag{16}$$

$$FDR = \frac{FP}{TP + FP} \tag{17}$$

where TP is the number of correctly detected sample defects in the testing process, TN is the number of correctly detected true defect-free samples, FP is the number of incorrectly detected true defect-free samples as defective samples, and FN is the number of incorrectly judged defects as true sample backgrounds. The IOU of the detection model in the testing process is set to 0.25. The results of the validation set of (a)–(j) are shown in Tables 4–14.

Table 4. Indicators for each detection model in De-Test A.

Model	**YOLOv5**			**SSD**			**Faster-RCNN**		
	DDA	**DDR**	**FDR**	**DDA**	**DDR**	**FDR**	**DDA**	**DDR**	**FDR**
Class 1	88.89%	66.67%	0.00%	81.67%	46.67%	3.45%	87.78%	63.33%	0.00%
Class 2	93.89%	81.67%	0.00%	92.22%	76.67%	0.00%	91.67%	75.00%	0.00%
Class 3	89.44%	73.33%	6.38%	88.33%	71.67%	8.51%	90.56%	75.00%	4.26%
Class 4	79.44%	46.67%	15.15%	81.11%	50.00%	11.76%	80.56%	48.33%	12.12%
Class 5	86.67%	68.33%	10.87%	85.00%	66.67%	14.89%	86.11%	68.33%	12.77%
Class 6	90.56%	73.33%	2.22%	88.89%	70.00%	4.55%	89.44%	71.67%	4.44%
Class 7	88.89%	71.67%	6.52%	87.78%	70.00%	8.70%	88.89%	71.67%	6.52%
Class 8	84.44%	65.00%	15.22%	81.11%	58.33%	20.45%	84.44%	65.00%	15.22%
Class 9	83.89%	68.33%	19.61%	84.44%	70.00%	19.23%	85.56%	71.67%	17.31%
Class 10	92.22%	76.67%	0.00%	92.78%	78.33%	0.00%	92.22%	76.67%	0.00%

Table 5. Class 1 test result.

Model	**Train Dataset**	**Test Dataset**	**De-Test B**			**De-Test C**		
			DDA	**DDR**	**FDR**	**DDA**	**DDR**	**FDR**
YOLOv5	De-Train B	Tree-CycleGAN [42]	88.46%	69.35%	4.44%	90.06%	73.77%	4.26%
		CycleGAN-TSS [43]	88.40%	68.85%	4.55%	90.61%	75.41%	4.17%
		PreCaCycleGAN	**89.62%**	**71.43%**	**2.17%**	**92.86%**	**80.65%**	**1.96%**
	De-Train C	Tree-CycleGAN [42]	90.11%	72.58%	2.17%	92.27%	78.69%	2.04%
		CycleGAN-TSS [43]	91.21%	74.19%	0.00%	93.37%	80.33%	0.00%
		PreCaCycleGAN	**91.89%**	**76.92%**	**0.00%**	**94.54%**	**84.13%**	**0.00%**

Table 5. *Cont.*

Model	**Train Dataset**	**Test Dataset**	**De-Test B**			**De-Test C**		
			DDA	**DDR**	**FDR**	**DDA**	**DDR**	**FDR**
SSD	De-Train B	Tree-CycleGAN [42]	81.87%	51.61%	8.57%	82.32%	52.46%	8.57%
		CycleGAN-TSS [43]	82.87%	52.46%	5.88%	83.43%	54.10%	5.71%
		PreCaCycleGAN	**85.25%**	**58.73%**	**2.63%**	**85.71%**	**59.68%**	**2.63%**
	De-Train C	Tree-CycleGAN [42]	83.52%	53.23%	2.94%	84.53%	55.74%	2.86%
		CycleGAN-TSS [43]	84.07%	54.84%	2.86%	85.08%	57.38%	2.78%
		PreCaCycleGAN	**86.49%**	**61.54%**	**0.00%**	**87.43%**	**63.49%**	**0.00%**
Faster-RCNN	De-Train B	Tree-CycleGAN [42]	87.36%	66.13%	4.65%	88.40%	68.85%	4.55%
		CycleGAN-TSS [43]	88.40%	68.85%	4.55%	88.95%	70.49%	4.44%
		PreCaCycleGAN	**90.16%**	**73.02%**	**2.13%**	**90.66%**	**74.19%**	**2.13%**
	De-Train C	Tree-CycleGAN [42]	88.46%	67.74%	2.33%	89.50%	70.49%	2.27%
		CycleGAN-TSS [43]	89.56%	69.35%	0.00%	90.61%	72.13%	0.00%
		PreCaCycleGAN	**90.81%**	**73.85%**	**0.00%**	**92.35%**	**77.78%**	**0.00%**

Table 6. Class 2 test result.

Model	**Train Dataset**	**Test Dataset**	**De-Test B**			**De-Test C**		
			DDA	**DDR**	**FDR**	**DDA**	**DDR**	**FDR**
YOLOv5	De-Train B	Tree-CycleGAN [42]	95.14%	86.15%	0.00%	96.15%	88.71%	0.00%
		CycleGAN-TSS [43]	94.54%	84.13%	0.00%	95.58%	86.89%	0.00%
		PreCaCycleGAN	**96.50%**	**91.25%**	**0.00%**	**96.86%**	**91.55%**	**0.00%**
	De-Train C	Tree-CycleGAN [42]	96.26%	89.55%	0.00%	97.81%	93.65%	0.00%
		CycleGAN-TSS [43]	95.65%	87.50%	0.00%	97.24%	91.80%	0.00%
		PreCaCycleGAN	**97.52%**	**93.90%**	**0.00%**	**98.96%**	**97.26%**	**0.00%**
SSD	De-Train B	Tree-CycleGAN [42]	93.51%	81.54%	0.00%	94.51%	83.87%	0.00%
		CycleGAN-TSS [43]	92.35%	79.37%	1.96%	92.82%	80.33%	2.00%
		PreCaCycleGAN	**94.00%**	**85.00%**	**0.00%**	**94.76%**	**85.92%**	**0.00%**
	De-Train C	Tree-CycleGAN [42]	94.12%	83.58%	0.00%	95.08%	85.71%	0.00%
		CycleGAN-TSS [43]	93.48%	81.25%	0.00%	94.48%	83.61%	0.00%
		PreCaCycleGAN	**95.05%**	**87.80%**	**0.00%**	**95.34%**	**87.67%**	**0.00%**
Faster-RCNN	De-Train B	Tree-CycleGAN [42]	92.43%	80.00%	1.89%	93.41%	82.26%	1.92%
		CycleGAN-TSS [43]	91.80%	77.78%	2.00%	92.82%	80.33%	2.00%
		PreCaCycleGAN	**94.00%**	**85.00%**	**0.00%**	**94.76%**	**85.92%**	**0.00%**
	De-Train C	Tree-CycleGAN [42]	93.58%	82.09%	0.00%	94.54%	84.13%	0.00%
		CycleGAN-TSS [43]	92.93%	79.69%	0.00%	93.37%	80.33%	0.00%
		PreCaCycleGAN	**95.05%**	**87.80%**	**0.00%**	**95.85%**	**89.04%**	**0.00%**

Table 7. Class 3 test result.

Model	**Train Dataset**	**Test Dataset**	**De-Test B**			**De-Test C**		
			DDA	**DDR**	**FDR**	**DDA**	**DDR**	**FDR**
YOLOv5	De-Train B	Tree-CycleGAN [42]	95.14%	86.15%	0.00%	96.15%	88.71%	0.00%
		CycleGAN-TSS [43]	94.54%	84.13%	0.00%	95.58%	86.89%	0.00%
		PreCaCycleGAN	**96.50%**	**91.25%**	**0.00%**	**96.86%**	**91.55%**	**0.00%**
	De-Train C	Tree-CycleGAN [42]	96.26%	89.55%	0.00%	97.81%	93.65%	0.00%
		CycleGAN-TSS [43]	95.65%	87.50%	0.00%	97.24%	91.80%	0.00%
		PreCaCycleGAN	**97.52%**	**93.90%**	**0.00%**	**98.96%**	**97.26%**	**0.00%**

Table 7. *Cont.*

Model	Train Dataset	Test Dataset	De-Test B			De-Test C		
			DDA	DDR	FDR	DDA	DDR	FDR
SSD	De-Train B	Tree-CycleGAN [42]	93.51%	81.54%	0.00%	94.51%	83.87%	0.00%
		CycleGAN-TSS [43]	92.35%	79.37%	1.96%	92.82%	80.33%	2.00%
		PreCaCycleGAN	**94.00%**	**85.00%**	**0.00%**	**94.76%**	**85.92%**	**0.00%**
	De-Train C	Tree-CycleGAN [42]	94.12%	83.58%	0.00%	95.08%	85.71%	0.00%
		CycleGAN-TSS [43]	93.48%	81.25%	0.00%	94.48%	83.61%	0.00%
		PreCaCycleGAN	**95.05%**	**87.80%**	**0.00%**	**95.34%**	**87.67%**	**0.00%**
Faster-RCNN	De-Train B	Tree-CycleGAN [42]	92.43%	80.00%	1.89%	93.41%	82.26%	1.92%
		CycleGAN-TSS [43]	91.80%	77.78%	2.00%	92.82%	80.33%	2.00%
		PreCaCycleGAN	**94.00%**	**85.00%**	**0.00%**	**94.76%**	**85.92%**	**0.00%**
	De-Train C	Tree-CycleGAN [42]	93.58%	82.09%	0.00%	94.54%	84.13%	0.00%
		CycleGAN-TSS [43]	92.93%	79.69%	0.00%	93.37%	80.33%	0.00%
		PreCaCycleGAN	**95.05%**	**87.80%**	**0.00%**	**95.85%**	**89.04%**	**0.00%**

Table 8. Class 4 test result.

Model	Train Dataset	Test Dataset	De-Test B			De-Test C		
			DDA	DDR	FDR	DDA	DDR	FDR
YOLOv5	De-Train B	Tree-CycleGAN [42]	79.44%	50.00%	18.92%	80.56%	53.33%	17.95%
		CycleGAN-TSS [43]	80.56%	51.67%	16.22%	81.67%	55.00%	15.38%
		PreCaCycleGAN	**81.22%**	**54.10%**	**15.38%**	**81.77%**	**55.74%**	**15.00%**
	De-Train C	Tree-CycleGAN [42]	82.32%	52.46%	8.57%	83.43%	55.74%	8.11%
		CycleGAN-TSS [43]	84.07%	**56.45%**	5.41%	**84.53%**	**57.38%**	**5.41%**
		PreCaCycleGAN	**83.61%**	55.56%	**5.41%**	83.98%	55.74%	5.56%
SSD	De-Train B	Tree-CycleGAN [42]	81.67%	53.33%	13.51%	82.22%	55.00%	13.16%
		CycleGAN-TSS [43]	83.33%	56.67%	10.53%	83.89%	58.33%	10.26%
		PreCaCycleGAN	**83.43%**	**57.38%**	**10.26%**	**83.98%**	**59.02%**	**10.00%**
	De-Train C	Tree-CycleGAN [42]	84.53%	57.38%	5.41%	86.19%	62.30%	5.00%
		CycleGAN-TSS [43]	**85.71%**	**59.68%**	2.63%	86.74%	62.30%	2.56%
		PreCaCycleGAN	85.25%	58.73%	**2.63%**	**86.74%**	**62.30%**	**2.56%**
Faster-RCNN	De-Train B	Tree-CycleGAN [42]	80.56%	50.00%	14.29%	81.67%	53.33%	13.51%
		CycleGAN-TSS [43]	80.56%	50.00%	14.29%	81.67%	53.33%	13.51%
		PreCaCycleGAN	**81.77%**	**54.10%**	**13.16%**	**82.32%**	**55.74%**	**12.82%**
	De-Train C	Tree-CycleGAN [42]	83.43%	55.74%	8.11%	83.98%	57.38%	7.89%
		CycleGAN-TSS [43]	84.62%	**58.06%**	5.26%	85.08%	59.02%	5.26%
		PreCaCycleGAN	**84.70%**	57.14%	**2.70%**	**85.64%**	**59.02%**	**2.70%**

Table 9. Class 5 test result.

Model	Train Dataset	Test Dataset	De-Test B			De-Test C		
			DDA	DDR	FDR	DDA	DDR	FDR
YOLOv5	De-Train B	Tree-CycleGAN [42]	89.56%	74.19%	6.12%	90.61%	77.05%	6.00%
		CycleGAN-TSS [43]	88.95%	72.13%	6.38%	90.06%	75.41%	6.12%
		PreCaCycleGAN	**90.31%**	**77.63%**	**3.28%**	**91.19%**	**79.45%**	**3.33%**
	De-Train C	Tree-CycleGAN [42]	91.80%	77.78%	2.00%	92.82%	80.33%	2.00%
		CycleGAN-TSS [43]	90.81%	76.92%	3.85%	91.76%	79.03%	3.92%
		PreCaCycleGAN	**92.75%**	**82.76%**	**0.00%**	**93.72%**	**83.10%**	**0.00%**

Table 9. *Cont.*

Model	Train Dataset	Test Dataset	De-Test B			De-Test C		
			DDA	DDR	FDR	DDA	DDR	FDR
SSD	De-Train B	Tree-CycleGAN [42]	87.36%	70.97%	10.20%	88.40%	73.77%	10.00%
		CycleGAN-TSS [43]	86.19%	68.85%	12.50%	87.29%	72.13%	12.00%
		PreCaCycleGAN	**88.27%**	**76.32%**	**7.94%**	**89.12%**	**78.08%**	**8.06%**
	De-Train C	Tree-CycleGAN [42]	90.16%	74.60%	4.08%	90.61%	75.41%	4.17%
		CycleGAN-TSS [43]	89.19%	73.85%	5.88%	89.01%	72.58%	6.25%
		PreCaCycleGAN	**91.79%**	**81.61%**	**1.39%**	**91.10%**	**77.46%**	**1.79%**
Faster-RCNN	De-Train B	Tree-CycleGAN [42]	89.01%	72.58%	6.25%	90.06%	75.41%	6.12%
		CycleGAN-TSS [43]	87.85%	70.49%	8.51%	88.95%	73.77%	8.16%
		PreCaCycleGAN	**90.31%**	**77.63%**	**3.28%**	**90.67%**	**78.08%**	**3.39%**
	De-Train C	Tree-CycleGAN [42]	90.71%	76.19%	4.00%	91.71%	78.69%	4.00%
		CycleGAN-TSS [43]	90.27%	75.38%	3.92%	91.21%	77.42%	4.00%
		PreCaCycleGAN	**92.27%**	**82.76%**	**1.37%**	**92.15%**	**80.28%**	**1.72%**

Table 10. Class 6 test result.

Model	Train Dataset	Test Dataset	De-Test B			De-Test C		
			DDA	DDR	FDR	DDA	DDR	FDR
YOLOv5	De-Train B	Tree-CycleGAN [42]	91.85%	79.69%	3.77%	90.71%	80.95%	8.93%
		CycleGAN-TSS [43]	91.76%	79.03%	3.92%	92.86%	82.26%	3.77%
		PreCaCycleGAN	**92.90%**	**80.95%**	**1.92%**	**93.96%**	**83.87%**	**1.89%**
	De-Train C	Tree-CycleGAN [42]	93.01%	81.82%	1.82%	93.48%	84.38%	3.57%
		CycleGAN-TSS [43]	93.99%	82.54%	0.00%	95.05%	85.48%	0.00%
		PreCaCycleGAN	**95.85%**	**89.04%**	**0.00%**	**96.79%**	**91.04%**	**0.00%**
SSD	De-Train B	Tree-CycleGAN [42]	89.13%	73.44%	6.00%	88.52%	77.78%	12.50%
		CycleGAN-TSS [43]	90.11%	74.19%	4.17%	90.66%	77.42%	5.88%
		PreCaCycleGAN	**91.26%**	**76.19%**	**2.04%**	**92.31%**	**79.03%**	**2.00%**
	De-Train C	Tree-CycleGAN [42]	92.47%	80.30%	1.85%	91.30%	81.25%	7.14%
		CycleGAN-TSS [43]	91.80%	77.78%	2.00%	93.41%	80.65%	0.00%
		PreCaCycleGAN	**93.26%**	**82.19%**	**0.00%**	**94.12%**	**83.58%**	**0.00%**
Faster-RCNN	De-Train B	Tree-CycleGAN [42]	90.76%	76.56%	3.92%	89.62%	79.37%	10.71%
		CycleGAN-TSS [43]	91.21%	77.42%	4.00%	91.76%	79.03%	3.92%
		PreCaCycleGAN	**91.80%**	**77.78%**	**2.00%**	**92.86%**	**80.65%**	**1.96%**
	De-Train C	Tree-CycleGAN [42]	92.47%	80.30%	1.85%	92.39%	82.81%	5.36%
		CycleGAN-TSS [43]	93.44%	80.95%	0.00%	95.05%	85.48%	0.00%
		PreCaCycleGAN	**94.30%**	**84.93%**	**0.00%**	**95.19%**	**86.57%**	**0.00%**

Table 11. Class 7 test result.

Model	Train Dataset	Test Dataset	De-Test B			De-Test C		
			DDA	DDR	FDR	DDA	DDR	FDR
YOLOv5	De-Train B	Tree-CycleGAN [42]	91.21%	79.03%	5.77%	91.76%	80.65%	5.66%
		CycleGAN-TSS [43]	91.76%	79.03%	3.92%	92.31%	80.65%	3.85%
		PreCaCycleGAN	**92.93%**	**81.25%**	**1.89%**	**93.99%**	**84.13%**	**1.85%**
	De-Train C	Tree-CycleGAN [42]	92.82%	80.33%	2.00%	94.48%	85.25%	1.89%
		CycleGAN-TSS [43]	93.44%	80.95%	0.00%	95.05%	85.48%	0.00%
		PreCaCycleGAN	**95.19%**	**86.57%**	**0.00%**	**96.22%**	**89.23%**	**0.00%**

Table 11. *Cont.*

Model	Train Dataset	Test Dataset	De-Test B			De-Test C		
			DDA	DDR	FDR	DDA	DDR	FDR
SSD	De-Train B	Tree-CycleGAN [42]	89.01%	74.19%	8.00%	89.56%	75.81%	7.84%
		CycleGAN-TSS [43]	89.56%	74.19%	6.12%	90.11%	75.81%	6.00%
		PreCaCycleGAN	**91.30%**	**78.13%**	**3.85%**	**91.80%**	**79.37%**	**3.85%**
	De-Train C	Tree-CycleGAN [42]	91.16%	75.41%	2.13%	92.82%	80.33%	2.00%
		CycleGAN-TSS [43]	91.26%	76.19%	2.04%	92.86%	80.65%	1.96%
		PreCaCycleGAN	**93.05%**	**80.60%**	**0.00%**	**94.05%**	**83.08%**	**0.00%**
Faster-RCNN	De-Train B	Tree-CycleGAN [42]	91.76%	80.65%	5.66%	92.31%	82.26%	5.56%
		CycleGAN-TSS [43]	91.76%	79.03%	3.92%	92.86%	82.26%	3.77%
		PreCaCycleGAN	**92.93%**	**81.25%**	**1.89%**	**93.99%**	**84.13%**	**1.85%**
	De-Train C	Tree-CycleGAN [42]	93.37%	81.97%	1.96%	95.03%	86.89%	1.85%
		CycleGAN-TSS [43]	93.99%	82.54%	0.00%	95.60%	87.10%	0.00%
		PreCaCycleGAN	**95.19%**	**86.57%**	**0.00%**	**96.22%**	**89.23%**	**0.00%**

Table 12. Class 8 test result.

Model	Train Dataset	Test Dataset	De-Test B			De-Test C		
			DDA	DDR	FDR	DDA	DDR	FDR
YOLOv5	De-Train B	Tree-CycleGAN [42]	87.63%	75.76%	12.28%	88.04%	76.56%	12.50%
		CycleGAN-TSS [43]	86.34%	73.02%	14.81%	86.89%	74.60%	14.55%
		PreCaCycleGAN	**87.75%**	**79.76%**	**10.67%**	**88.32%**	**80.52%**	**11.43%**
	De-Train C	Tree-CycleGAN [42]	90.43%	77.94%	5.36%	**90.81%**	78.46%	**5.56%**
		CycleGAN-TSS [43]	88.17%	74.24%	9.26%	89.13%	76.56%	9.26%
		PreCaCycleGAN	**91.24%**	**84.54%**	**4.65%**	90.55%	**81.48%**	5.71%
SSD	De-Train B	Tree-CycleGAN [42]	82.80%	68.18%	19.64%	83.15%	68.75%	20.00%
		CycleGAN-TSS [43]	81.97%	65.08%	21.15%	83.06%	68.25%	20.37%
		PreCaCycleGAN	**85.29%**	**76.19%**	**13.51%**	**85.28%**	**75.32%**	**14.71%**
	De-Train C	Tree-CycleGAN [42]	86.70%	72.06%	10.91%	87.03%	72.31%	11.32%
		CycleGAN-TSS [43]	85.48%	69.70%	13.21%	85.87%	70.31%	13.46%
		PreCaCycleGAN	**88.02%**	**79.38%**	**7.23%**	**88.06%**	**77.78%**	**8.70%**
Faster-RCNN	De-Train B	Tree-CycleGAN [42]	86.02%	72.73%	14.29%	86.41%	73.44%	14.55%
		CycleGAN-TSS [43]	85.79%	71.43%	15.09%	86.34%	73.02%	14.81%
		PreCaCycleGAN	**87.25%**	**78.57%**	**10.81%**	**87.31%**	**77.92%**	**11.76%**
	De-Train C	Tree-CycleGAN [42]	88.83%	75.00%	7.27%	89.73%	76.92%	7.41%
		CycleGAN-TSS [43]	88.71%	74.24%	7.55%	89.67%	76.56%	7.55%
		PreCaCycleGAN	**90.32%**	**82.47%**	**4.76%**	**90.55%**	**81.48%**	**5.71%**

Table 13. Class 9 test result.

Model	Train Dataset	Test Dataset	De-Test B			De-Test C		
			DDA	DDR	FDR	DDA	DDR	FDR
YOLOv5	De-Train B	Tree-CycleGAN [42]	84.49%	73.13%	18.33%	85.48%	75.76%	18.03%
		CycleGAN-TSS [43]	84.78%	73.44%	18.97%	85.71%	75.81%	18.97%
		PreCaCycleGAN	**87.08%**	**82.02%**	**13.10%**	**87.06%**	**81.48%**	**14.29%**
	De-Train C	Tree-CycleGAN [42]	88.89%	78.26%	10.00%	90.32%	81.82%	10.00%
		CycleGAN-TSS [43]	88.71%	77.27%	10.53%	89.67%	79.69%	10.53%
		PreCaCycleGAN	**91.03%**	**86.41%**	**6.32%**	**91.22%**	**85.88%**	**7.59%**

Table 13. *Cont.*

Model	Train Dataset	Test Dataset	De-Test B			De-Test C		
			DDA	DDR	FDR	DDA	DDR	FDR
SSD	De-Train B	Tree-CycleGAN [42]	86.10%	77.61%	17.46%	86.56%	78.79%	17.46%
		CycleGAN-TSS [43]	85.33%	75.00%	18.64%	86.26%	77.42%	18.64%
		PreCaCycleGAN	**87.56%**	**83.15%**	**12.94%**	**87.56%**	**82.72%**	**14.10%**
	De-Train C	Tree-CycleGAN [42]	89.95%	81.16%	9.68%	90.32%	81.82%	10.00%
		CycleGAN-TSS [43]	89.25%	78.79%	10.34%	90.22%	81.25%	10.34%
		PreCaCycleGAN	**91.48%**	**87.38%**	**6.25%**	**91.71%**	**87.06%**	**7.50%**
Faster-RCNN	De-Train B	Tree-CycleGAN [42]	88.24%	82.09%	15.38%	88.71%	83.33%	15.38%
		CycleGAN-TSS [43]	88.04%	81.25%	16.13%	88.46%	82.26%	16.39%
		PreCaCycleGAN	**89.47%**	**86.52%**	**11.49%**	**90.05%**	**87.65%**	**12.35%**
	De-Train C	Tree-CycleGAN [42]	92.06%	85.51%	7.81%	91.94%	84.85%	8.20%
		CycleGAN-TSS [43]	91.94%	84.85%	8.20%	92.39%	85.94%	8.33%
		PreCaCycleGAN	**93.27%**	**90.29%**	**5.10%**	**93.66%**	**90.59%**	**6.10%**

Table 14. Class 10 test result.

Model	Train Dataset	Test Dataset	De-Test B			De-Test C		
			DDA	DDR	FDR	DDA	DDR	FDR
YOLOv5	De-Train B	Tree-CycleGAN [42]	92.86%	80.65%	1.96%	93.92%	83.61%	1.92%
		CycleGAN-TSS [43]	92.43%	80.00%	1.89%	93.41%	82.26%	1.92%
		PreCaCycleGAN	**94.47%**	**86.08%**	**0.00%**	**94.71%**	**85.51%**	**0.00%**
	De-Train C	Tree-CycleGAN [42]	94.57%	84.38%	0.00%	95.03%	85.25%	0.00%
		CycleGAN-TSS [43]	94.09%	83.33%	0.00%	95.08%	85.71%	0.00%
		PreCaCycleGAN	**95.02%**	**87.65%**	**0.00%**	**95.81%**	**88.73%**	**0.00%**
SSD	De-Train B	Tree-CycleGAN [42]	92.86%	80.65%	1.96%	93.37%	81.97%	1.96%
		CycleGAN-TSS [43]	94.05%	83.08%	0.00%	94.51%	83.87%	0.00%
		PreCaCycleGAN	**94.47%**	**86.08%**	**0.00%**	**94.71%**	**85.51%**	**0.00%**
	De-Train C	Tree-CycleGAN [42]	94.57%	84.38%	0.00%	95.58%	86.89%	0.00%
		CycleGAN-TSS [43]	94.62%	84.85%	0.00%	95.63%	87.30%	0.00%
		PreCaCycleGAN	**95.52%**	**88.89%**	**0.00%**	**96.34%**	**90.14%**	**0.00%**
Faster-RCNN	De-Train B	Tree-CycleGAN [42]	92.86%	80.65%	1.96%	93.92%	83.61%	1.92%
		CycleGAN-TSS [43]	93.51%	83.08%	1.82%	93.96%	83.87%	1.89%
		PreCaCycleGAN	**93.97%**	**84.81%**	**0.00%**	**94.71%**	**85.51%**	**0.00%**
	De-Train C	Tree-CycleGAN [42]	94.57%	84.38%	0.00%	95.03%	85.25%	0.00%
		CycleGAN-TSS [43]	94.62%	84.85%	0.00%	95.08%	85.71%	0.00%
		PreCaCycleGAN	**95.02%**	**87.65%**	**0.00%**	**95.29%**	**87.32%**	**0.00%**

4.3. Discussion

Out of 360 test results consisting of ten datasets, three generation models, and three detection models, our algorithm achieved 354 optimal results. Overall, the detection models trained with our model-generated defects mixed with real defects, YOLOv5 detection models in ten datasets compared to the original detection models, CycleGAN-TSS generation models and Tree-CycleGAN generation models improved the detection accuracy by 5.75%, 1.16% and 1.26% on average, respectively, and the average improvement in detection rate by an average of 14.94%, 3.60% and 3.55% improvement, and 5.44%, 1.22% and 1.63% decrease in false detection rate; SSD detection model compared to the original detection model, CycleGAN-TSS generation model and Tree-CycleGAN generation model detection accuracy improved by 5.46%, 1.23% and 1.27% on average, respectively, with an average improvement of detection rate by 13.73%, 3.76% and 3.35% on average, and the false detection rate by 6.74%, 1.89% and 2.42%; Faster-RCNN detection model compared to the original detection model, CycleGAN-TSS generation model and Tree-CycleGAN generation model detection accuracy by 5.64%, on average, respectively 0.88% and 1.42%, the average improvement in detection rate is 14.54%, 3.22% and 3.62% on average, and the false detection rate is decreased by 5.64%, 0.89% and 2.26%.

The test results show that compared with the two generation models of Tree-CycleGAN and CycleGAN-TSS, our model can extract the local and global features of defects more effectively and improve the fineness of features by adding perceptual functions and capsule discriminators for (d) and (i), which are datasets with complex backgrounds and obscure performance of defective features, Class4 and Class9 detection results are shown in Figures 5 and 6. The false detection rate of the original Faster-RCNN model in (d) is 12.12%, the false detection rate of the Tree-CycleGAN model that detects mixed samples after mixed sample training is 7.89%, and the false detection rate of the CycleGAN-TSS model is 5.26%, and our model can reduce the false detection rate to 2.70%; in (i) The false detection rate of the original YOLOv5 model is 19.61%, and the false detection rate of the Tree-CycleGAN model that detects mixed samples after mixed sample training is 10.00%, and the false detection rate of the CycleGAN-TSS model is 10.53%, and our model can reduce the false detection rate to 7.59%.

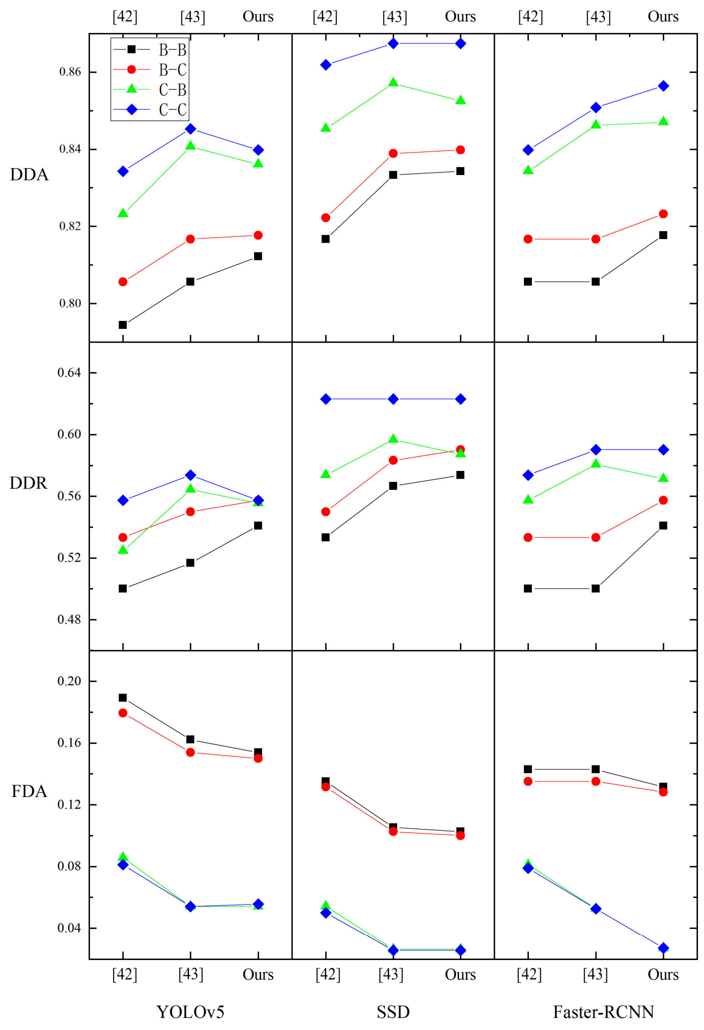

Figure 5. Class 4 test results, Wherein B-B stands for the meaning of the results tested on model DE-Test B trained using DE-Train B, B-C stands for the meaning of the results tested on model DE-Test C trained using DE-Train B, C-B stands for the meaning of the results tested on model DE-Test B trained using DE-Train C, and C-C stands for the meaning of the results tested on model DE-Test C trained using DE-Train C.

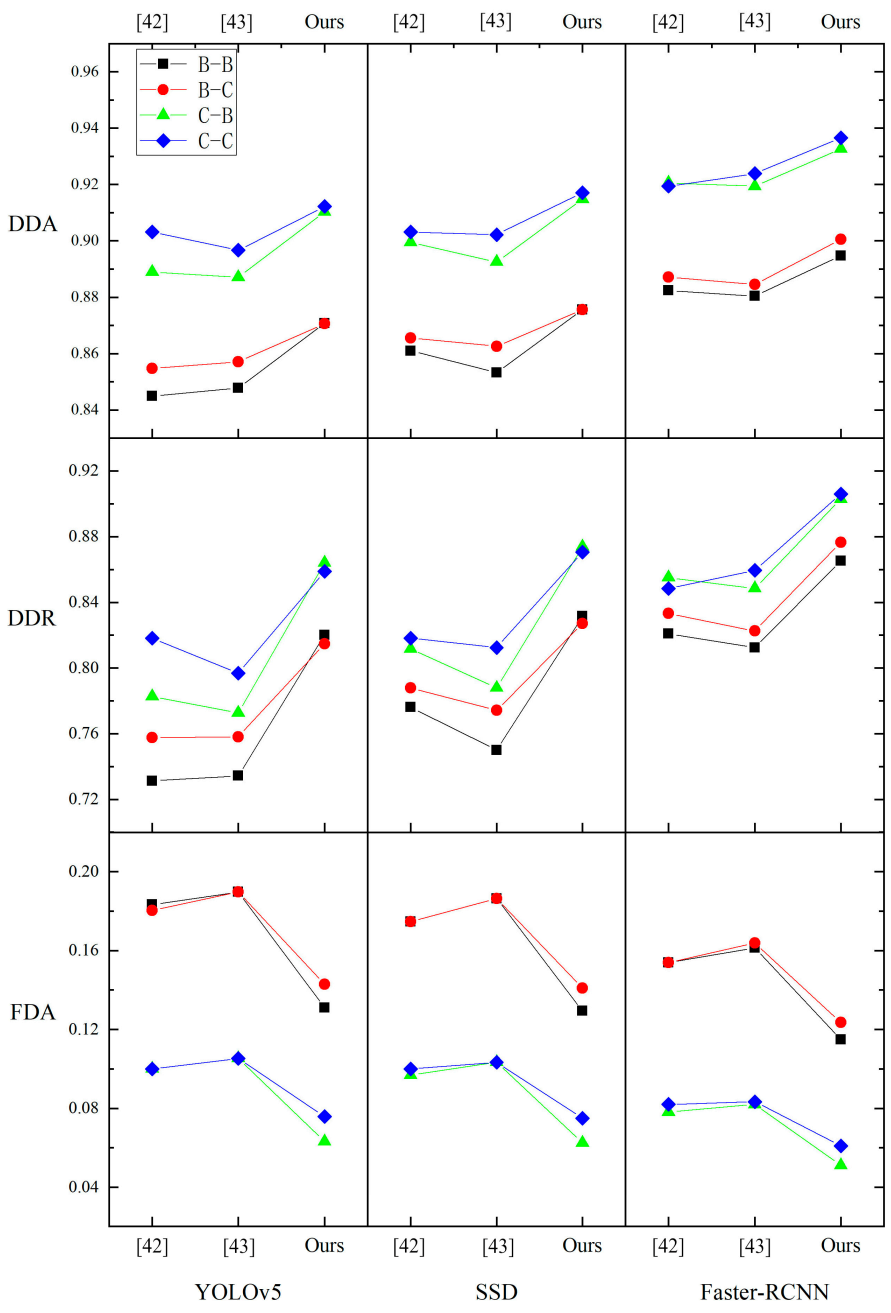

Figure 6. Class 9 test results.

For datasets with obvious defect features like (b) and (f), our model combines U-Net and DenseNet to enhance the defect feature representation of the samples and improve the learning ability of the defect detection model for defect features, Class2 and Class6 detection results are shown in Figures 7 and 8. the detection rate of the original YOLOv5 model in (b) is 81.67%, and the detection of the mixed samples after training of the mixed samples Tree. The detection rate of the original YOLOv5 model in (f) is 73.33%, and the detection rate of the Tree-CycleGAN model for detecting mixed samples after training of mixed samples is 91.80%, and our model can improve the detection rate to 97.26%, with a detection rate of 84.38%, and the CycleGAN-TSS model with a false detection rate of 85.48%, our model was able to improve the detection rate to 91.04%.

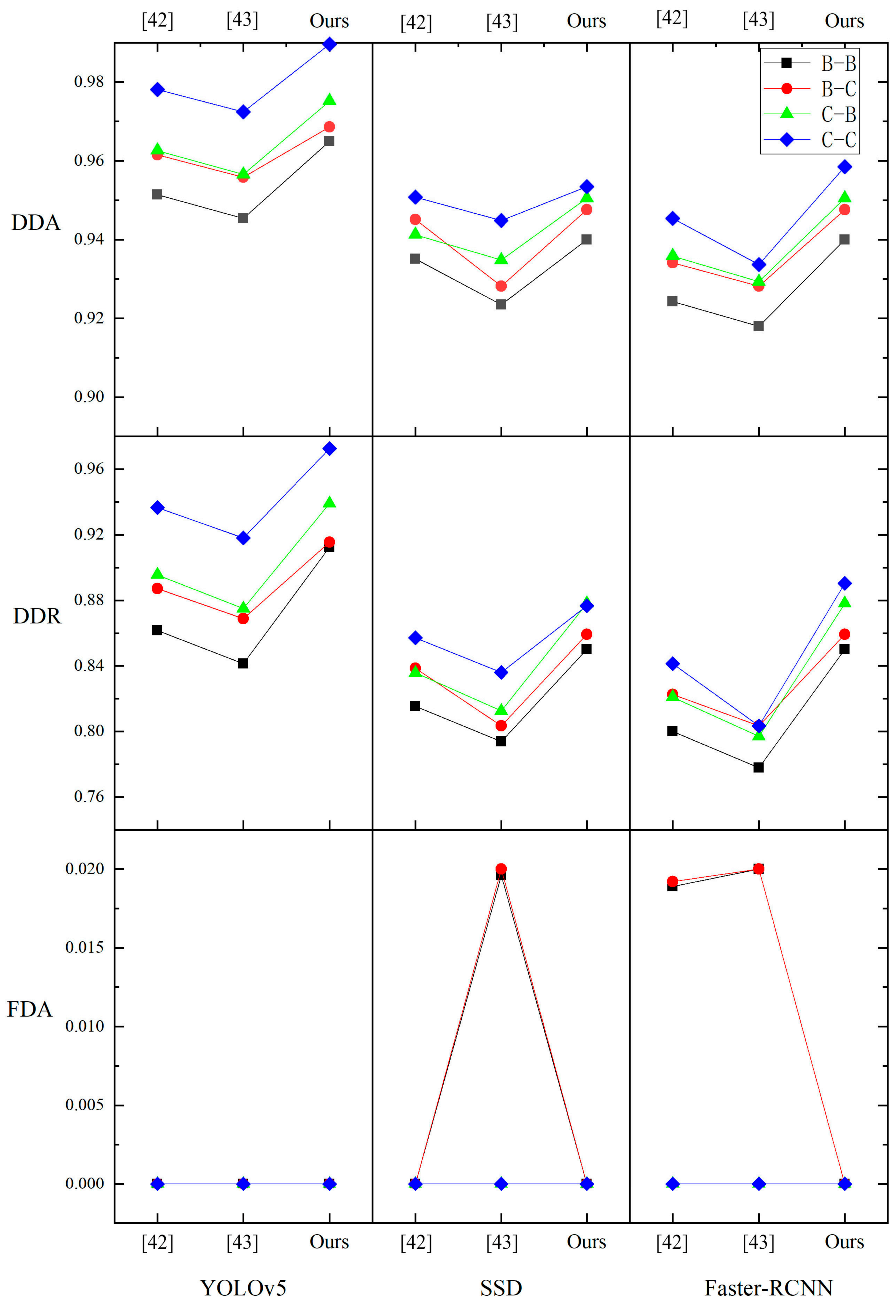

Figure 7. Class 2 test results.

The comparison experiments show that our model can generate high-quality defect samples on different types of defect datasets, and mixing and matching real defect samples can further improve the generalization ability and robustness of the defect detection model, which can effectively identify defects with obscure features in complex backgrounds and further enhance the authenticity and diversity of defect features.

Subsequent IOU value optimization for our optimal model for actual industrial defect detection, and finally, when the IOU value is 0.15, the false detection rate is 0%, and the average accuracy rate of various data sets reaches 98.73%. The experiments show that the defect samples generated by our model are better than those generated by the current defect sample generation model and can be practically applied to industrial defect detection, which can effectively improve the robustness and generalization of the defect detection model.

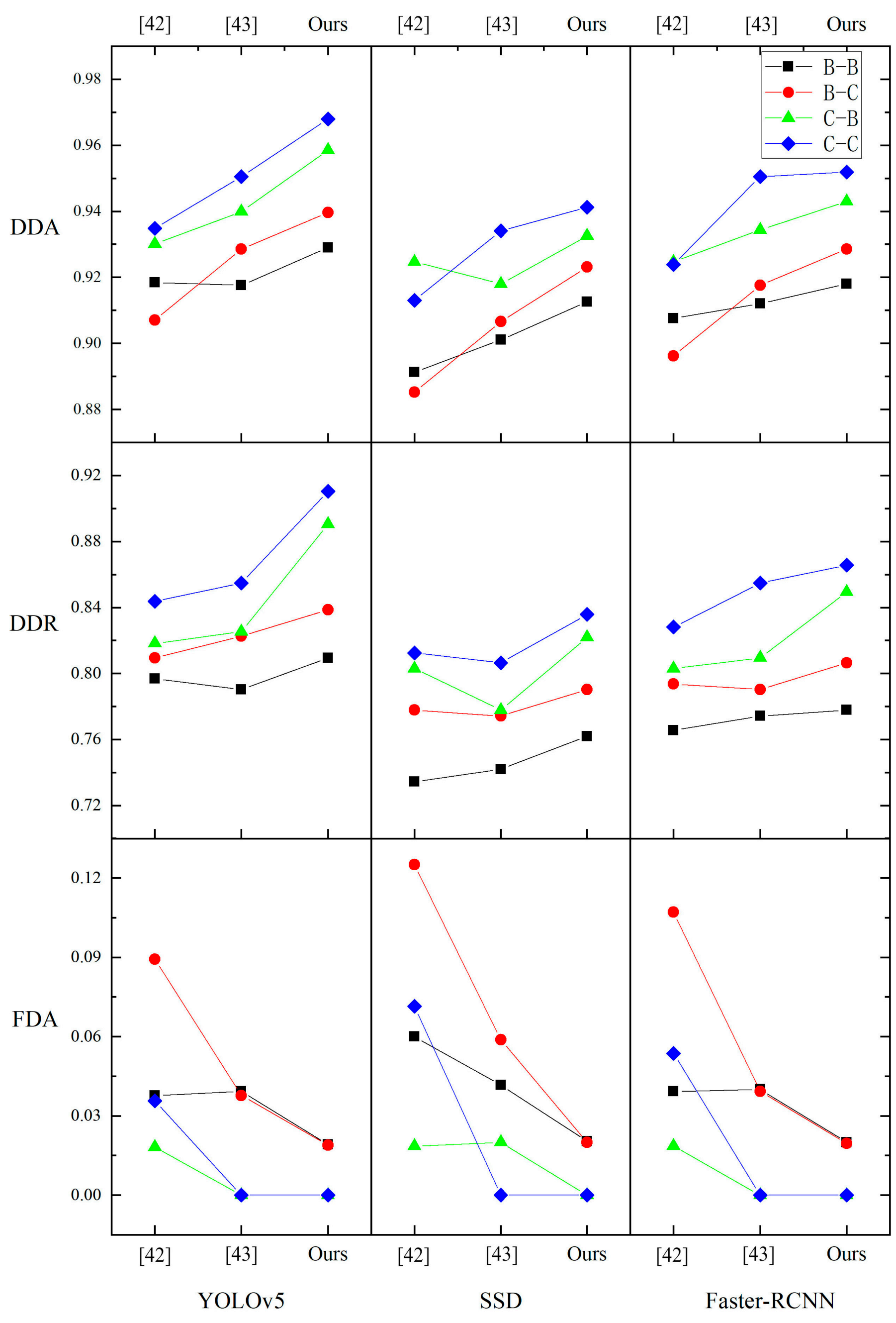

Figure 8. Class 6 test results.

5. Conclusions

In this work, we design PreCaCycleGAN, a model-based image-enhanced adversarial generative network for industrial defect sample augmentation, which aims to address the problem of defect data scarcity in industrial defect detection by current deep learning methods and to improve the generalization and robustness of the defect detection model in real industrial manufacturing. We show that our model uses the DenseNet module to enhance the U-Net network to synthesize defect samples and that the defect samples synthesized on different datasets are more realistic and faithful to the defect samples. We also show that our model uses a dual-branch discriminator and an added damage function to make the defect details more refined. We compare the augmentation effect of PreCaCycleGAN on different detection models with the state-of-the-art industrial defect sample generation models and demonstrate that our model has better defect sample generation diversity. However, our proposed method still has some limitations in practical industrial defect detection applications. On the one hand, it is not very effective in some industrial scenarios with limited resources or high-efficiency requirements. On the other hand, the generated industrial defect samples still need manual annotation. In the future, we will reduce the algorithm complexity to further lower the computational overhead in defect sample generation, and integrate the annotation algorithm to achieve automatic annotation of the generated defect samples, so as to better adapt to the industrial defect detection application scenarios.

Author Contributions: Formal analysis, Y.Y. and H.Z.; Investigation, Y.Y. and H.Z.; Resources, K.W.; Writing—original draft, J.Y.; Writing—review & editing, F.L., Y.Y. and H.Z.; Supervision, Y.Y. and H.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the General Program of Chongqing Natural Science Foundation grant number No. CSTB2022NSCQ-MSX0295.

Data Availability Statement: Data sets generated during the current study are available from the corresponding author upon reasonable request. The DAGM2007 public dataset can be found at https://hci.iwr.uni-heidelberg.de/content/weakly-supervised-learning-industrial-optical-inspection (accessed on 20 December 2022).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Javaid, M.; Haleem, A.; Singh, R.P.; Suman, R.; Gonzalez, E.S. Understanding the adoption of Industry 4.0 technologies in improving environmental sustainability. *Sustain. Oper. Comput.* **2022**, *3*, 203–217. [CrossRef]
2. Sofic, A.; Rakic, S.; Pezzotta, G.; Markoski, B.; Arioli, V.; Marjanovic, U. Smart and Resilient Transformation of Manufacturing Firms. *Processes* **2022**, *10*, 2674. [CrossRef]
3. Bastos, T.; Salvadorinho, J.; Teixeira, L. UpSkill@ Mgmt 4.0-A digital tool for competence management: Conceptual model and a prototype. *Int. J. Ind. Eng. Manag.* **2022**, *13*, 225–238. [CrossRef]
4. Zhou, L.; Zhang, L.; Konz, N. Computer vision techniques in manufacturing. *IEEE Trans. Syst. Man Cybern. Syst.* **2022**, *53*, 105–117. [CrossRef]
5. Ren, Z.; Fang, F.; Yan, N.; Wu, Y. State of the art in defect detection based on machine vision. *Int. J. Precis. Eng. Manuf. Green Technol.* **2022**, *9*, 661–691. [CrossRef]
6. Chen, Z.; Deng, J.; Zhu, Q.; Wang, H.; Chen, Y. A Systematic Review of Machine-Vision-Based Leather Surface Defect Inspection. *Electronics* **2022**, *11*, 2383. [CrossRef]
7. Kahraman, Y.; Durmuşoğlu, A. Deep learning-based fabric defect detection: A review. *Text. Res. J.* **2023**, *93*, 1485–1503. [CrossRef]
8. Li, Z.; Lin, H.; Liu, Y.; Chen, C.; Xia, Y. An industrial defect detection algorithm based on CPU-GPU parallel call. *Multimed. Tools Appl.* **2023**. [CrossRef]
9. Chen, C.; Abdullah, A.; Kok, S.H.; Tien, D.T.K. Review of industry workpiece classification and defect detection using deep learning. *Int. J. Adv. Comput. Sci. Appl.* **2022**, *13*, 4. [CrossRef]
10. Tulbure, A.A.; Tulbure, A.A.; Dulf, E.H. A review on modern defect detection models using DCNNs–Deep convolutional neural networks. *J. Adv. Res.* **2022**, *35*, 33–48. [CrossRef]
11. Jin, Q.; Chen, L. A survey of surface defect detection of industrial products based on a small number of labeled data. *arXiv* **2022**, arXiv:2203.05733.
12. Martin, D.; Heinzel, S.; von Bischhoffshausen, J.K.; Kühl, N. Deep learning strategies for industrial surface defect detection systems. *arXiv* **2021**, arXiv:2109.11304.
13. Yang, W. A Survey of Surface Defect Detection Based on Deep Learning. In Proceedings of the 2022 7th International Conference on Modern Management and Education Technology (MMET 2022), Xiamen, China, 23–25 September 2022; Atlantis Press: Amsterdam, The Netherlands, 2022; pp. 362–367.
14. Mirza, M.; Osindero, S. Conditional generative adversarial nets. *arXiv* **2014**, arXiv:1411.1784.
15. Odena, A.; Olah, C.; Shlens, J. Conditional image synthesis with auxiliary classifier gans. In Proceedings of the 34th International Conference on Machine Learning, PMLR, Sydney, Australia, 6–11 August 2017; pp. 2642–2651.
16. Arjovsky, M.; Chintala, S.; Bottou, L. Wasserstein generative adversarial networks. In Proceedings of the 34th International Conference on Machine Learning, PMLR, Sydney, Australia, 6–11 August 2017; pp. 214–223.
17. Isola, P.; Zhu, J.Y.; Zhou, T.; Efros, A.A. Image-to-image translation with conditional adversarial networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1125–1134.
18. Choi, Y.; Choi, M.; Kim, M.; Ha, J.W.; Kim, S.; Choo, J. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 8789–8797.

19. Ronneberger, O.; Fischer, P.; Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, 5–9 October 2015*; Proceedings, Part III 18; Springer International Publishing: Berlin/Heidelberg, Germany, 2015; pp. 234–241.
20. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4700–4708.
21. Zhu, J.Y.; Park, T.; Isola, P.; Efros, A.A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2223–2232.
22. Sabour, S.; Frosst, N.; Hinton, G.E. Dynamic routing between capsules. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Volume 30.
23. Yang, S.; Xiao, W.; Zhang, M.; Guo, S.; Zhao, J.; Shen, F. Image data augmentation for deep learning: A survey. *arXiv* **2022**, arXiv:2204.08610.
24. Yun, S.; Han, D.; Oh, S.J.; Chun, S.; Choe, J.; Yoo, Y. Cutmix: Regularization strategy to train strong classifiers with localizable features. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 6023–6032.
25. Zhong, Z.; Zheng, L.; Kang, G.; Li, S.; Yang, Y. Random erasing data augmentation. In Proceedings of the AAAI conference on artificial intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 13001–13008.
26. Moreno-Barea, F.J.; Strazzera, F.; Jerez, J.M.; Urda, D.; Franco, L. Forward noise adjustment scheme for data augmentation. In Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence (SSCI), Bengaluru, India, 18–21 November 2018; pp. 728–734.
27. Ghiasi, G.; Cui, Y.; Srinivas, A.; Qian, R.; Lin, T.Y.; Cubuk, E.D.; Le, Q.V.; Zoph, B. Simple copy-paste is a strong data augmentation method for instance segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 2918–2928.
28. Farady, I.; Lin, C.Y.; Chang, M.C. PreAugNet: Improve data augmentation for industrial defect classification with small-scale training data. *J. Intell. Manuf.* **2023**, 1–14. [CrossRef]
29. Cubuk, E.D.; Zoph, B.; Mane, D.; Vasudevan, V.; Le, Q.V. Autoaugment: Learning augmentation policies from data. *arXiv* **2018**, arXiv:1805.09501.
30. Cubuk, E.D.; Zoph, B.; Shlens, J.; Le, Q.V. Randaugment: Practical automated data augmentation with a reduced search space. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Seattle, WA, USA, 14–19 June 2020; pp. 702–703.
31. Zoph, B.; Le, Q.V. Neural architecture search with reinforcement learning. *arXiv* **2016**, arXiv:1611.01578.
32. Lim, S.; Kim, I.; Kim, T.; Kim, C.; Kim, S. Fast autoaugment. *Adv. Neural Inf. Process. Syst.* **2019**, *32*. [CrossRef]
33. Ho, D.; Liang, E.; Chen, X.; Stoica, I.; Abbeel, P. Population based augmentation: Efficient learning of augmentation policy schedules. In Proceedings of the 36th International Conference on Machine Learning, PMLR, Long Beach, CA, USA, 9–15 June 2019; pp. 2731–2741.
34. Zhang, X.; Wang, Q.; Zhang, J.; Zhong, Z. Adversarial autoaugment. *arXiv* **2019**, arXiv:1912.11188.
35. Kuo, C.W.; Ma, C.Y.; Huang, J.B.; Kira, Z. Featmatch: Feature-based augmentation for semi-supervised learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, 23–28 August 2020*; Proceedings, Part XVIII 16; Springer International Publishing: Berlin/Heidelberg, Germany, 2020; pp. 479–495.
36. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
37. Wong, S.C.; Gatt, A.; Stamatescu, V.; McDonnell, M.D. Understanding data augmentation for classification: When to warp? In Proceedings of the 2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA), Gold Coast, Australia, 30 November–2 December 2016; pp. 1–6.
38. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial networks. *Commun. ACM* **2020**, *63*, 139–144. [CrossRef]
39. Mao, X.; Li, Q.; Xie, H.; Lau, R.Y.K.; Wang, Z.; Smolley, S.P. Least squares generative adversarial networks. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2794–2802.
40. Nowozin, S.; Cseke, B.; Tomioka, R. f-gan: Training generative neural samplers using variational divergence minimization. In Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; Volume 29.
41. Chen, X.; Duan, Y.; Houthooft, R.; Schulman, J.; Sutskever, I.; Abbeel, P. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; Volume 29.
42. Qin, Y.; Wang, Z.; Xi, D. Tree CycleGAN with maximum diversity loss for image augmentation and its application into gear pitting detection. *Appl. Soft Comput.* **2022**, *114*, 108130. [CrossRef]
43. Song, J.; Li, P.; Fang, Q.; Xia, H.; Guo, R. Data Augmentation by an Additional Self-Supervised CycleGAN-Based for Shadowed Pavement Detection. *Sustainability* **2022**, *14*, 14304. [CrossRef]
44. Niu, S.; Lin, H.; Niu, T.; Li, B.; Wang, X. DefectGAN: Weakly-supervised defect detection using generative adversarial network. In Proceedings of the 2019 IEEE 15th International Conference on Automation Science and Engineering (CASE), Vancouver, BC, Canada, 22–26 August 2019; IEEE: Piscataway Township, NJ, USA; pp. 127–132.

45. Shao, G.; Huang, M.; Gao, F.; Liu, T.; Li, L. DuCaGAN: Unified dual capsule generative adversarial network for unsupervised image-to-image translation. *IEEE Access* **2020**, *8*, 154691–154707. [CrossRef]
46. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
47. Wieler, M.; Benchmark Dataset DAGM2007. German Association for Pattern Recognition and University of Heidelberg. 2007. Available online: https://hci.iwr.uni-heidelberg.de/content/weakly-supervised-learning-industrial-optical-inspection (accessed on 20 December 2022).
48. Jocher, G. Yolov5 [EB/0L].Code Repository. Available online: https://github.com/ultralytics/yolov5 (accessed on 20 December 2022).
49. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single shot multibox detector. In Computer Vision–ECCV 2016: In Proceedings of the 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016; Proceedings, Part I 14. Springer International Publishing: Berlin/Heidelberg, Germany, 2016; pp. 21–37.
50. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; Volume 28.
51. Li, Y.; Zhao, W.; Pan, J. Deformable patterned fabric defect detection with fisher criterion-based deep learning. *IEEE Trans. Autom. Sci. Eng.* **2016**, *14*, 1256–1264. [CrossRef]