agronomy

MDPI

Article

Lychee Surface Defect Detection Based on Deep Convolutional Neural Networks with GAN-Based Data Augmentation

Chenglong Wang [1] and Zhifeng Xiao [2,*]

[1] School of Electronic Information and Electrical Engineering, Huizhou University, Huizhou 516007, China; WadeBaileyoju@yahoo.com
[2] School of Engineering, Penn State Erie, The Behrend College, Erie, PA 16563, USA
* Correspondence: zux2@psu.edu; Tel.: +1-814-898-6252

Abstract: The performance of fruit surface defect detection is easily affected by factors such as noisy background and foliage occlusion. In this study, we choose lychee as a fruit type to investigate its surface quality. Lychees are hard to preserve and have to be stored at low temperatures to keep fresh. Additionally, the surface of lychees is subject to scratches and cracks during harvesting/processing. To explore the feasibility of the automation of defective surface detection for lychees, we build a dataset with 3743 samples divided into three categories, namely, mature, defects, and rot. The original dataset suffers an imbalanced distribution issue. To address it, we adopt a transformer-based generative adversarial network (GAN) as a means of data augmentation that can effectively enhance the original training set with more and diverse samples to rebalance the three categories. In addition, we investigate three deep convolutional neural network (DCNN) models, including SSD-MobileNet V2, Faster RCNN-ResNet50, and Faster RCNN-Inception-ResNet V2, trained under different settings for an extensive comparison study. The results show that all three models demonstrate consistent performance gains in mean average precision (mAP), with the application of GAN-based augmentation. The rebalanced dataset also reduces the inter-category discrepancy, allowing a DCNN model to be trained equally across categories. In addition, the qualitative results show that models trained under the augmented setting can better identify the critical regions and the object boundary, leading to gains in mAP. Lastly, we conclude that the most cost-effective model, SSD-MobileNet V2, presents a comparable mAP (91.81%) and a superior inference speed (102 FPS), suitable for real-time detection in industrial-level applications.

Keywords: lychee; deep convolutional neural network; generative adversarial network; surface defect detection; SSD; Faster RCNN; object detection

Citation: Wang, C.; Xiao, Z. Lychee Surface Defect Detection Based on Deep Convolutional Neural Networks with GAN-Based Data Augmentation. *Agronomy* **2021**, *11*, 1500. https://doi.org/10.3390/agronomy11081500

Academic Editors: Saeid Homayouni, Yacine Bouroubi and Karem Chokmani

Received: 25 June 2021
Accepted: 26 July 2021
Published: 28 July 2021

1. Introduction

The surface defects of fruits are one of the critical factors affecting their quality. Quickly and accurately identifying defective parts on a fruit's surface remains a challenging problem. Traditional fruit defect detection is carried out manually, which is costly, slow, and error prone. Additionally, the efficiency and accuracy of manual detection are affected by human fatigue. Therefore, there is an urgent need to develop an automated detection technology to boost fruit production and processing automation. At present, mechanical vision-based fruit surface defect detection and grading methods have been extensively studied. The commonly used methods are the region growing method, minimum outer rectangle method, threshold segmentation, edge detection, k-mean clustering segmentation, and contour finding [1–5]. However, the grayscale distribution of fruit surface damage is exceptionally irregular. The location, size, shape, and color of the damaged area are unpredictable. Uneven illumination and occlusion by obstacles also bring challenges to defect detection. Moreover, traditional image processing methods are limited by application types. In other words, a method suitable for one type of defect detection may not apply to a different type of defect.

Recent advances in deep learning have demonstrated the superiority of deep convolutional neural networks (DCNNs) in a broad spectrum of computer vision tasks such as object detection [6,7], image classification [8], segmentation [9], pose estimation [10], and autonomous driving [11], etc. A DCNN aims to learn and extract features from a raw image automatically, eliminating the step of feature design, which is usually carried out manually by someone with domain knowledge. In addition, the deep neural structure and the convolutional layers allow a model to extract high-quality and representative features efficiently, boosting model performance [12].

DCNN-based surface defect detection has gained extensive attention in recent years. Use cases have been mainly seen in manufacturing [13,14]. In the food/fruit industry, several prior efforts have employed DCNN models for surface defect and disease detection. Since the defects on different fruit/food present different features, most existing studies focus on building a detector for one type of fruit/food [15–22]. Zhu et al. developed an improved dense capsule network model for carrot grading [18]. Xie et al. conducted a comparative study on defective carrots detection using five DCNN models including Densenet-121, ResNet-50, Inception-V3, VGG-16, and VGG-19 [19]. A custom DCNN model was designed to detect defective mangosteen [20]. Tian et al. [23] employed a YOLOV3-Dense model with Cycle-Consistent GAN-based data augmentation for anthracnose lesion detection on the surface of apples. Da Costa et al. [21] investigated several DCNN models to detect external defects of tomatoes. Zhou et al. compared the VGG, and the ResNet18 models for a surface defects detection task of green plums [22]. To our best knowledge, research on lychee surface defect detection has not been explored.

Another challenge faced by DCNN-based surface defect detection is the lack of training samples, which are time-consuming and costly to obtain. A GAN, proposed by Ian Goodfellow in [24], is a class of neural network that is designed to generate data with the same distribution as the training samples. A GAN consists of two neural networks, including a generator and a discriminator. The former takes as input a random vector sampled from a latent space and generates a fake sample with the same dimension as a real sample. The latter is trained with the mixing of real and fake samples and predicts a binary value that indicates whether the input is real. The generator and the discriminator are trained alternatively to solve a MinMax game until both networks converge. Due to the strong ability of generative modeling, GANs are used for generative data augmentation, which can enhance the sample size and diversity of the dataset used for training, especially in the case of low resource and domain adaptations [25,26], where sufficient training data are unavailable. A notable effort is AugGAN [26], an image-to-image translation network consisting of encoders, generators, discriminators, and parsing nets for cross-domain adaptation. GANs are also extensively used in the biomedical field to generate high-quality medical images to train deep learning models. Use cases include skin [27] and liver [28] lesion classification, radiation therapy planning [29], and multi-contrast MRI generation [30,31]. A review of GANs in medical imaging is given in [32]. A review of GAN's application in manufacturing is given in [33]. Despite the numerous applications of GANs in data augmentation in fruit surface defect detection, the potential of GANs has not been fully explored. The only related study we have found is [23], in which the authors adopted CycleGAN to generate apple samples with anthracnose lesion.

In this paper, we aim to expand the study of fruit surface defect detection to lychees, a popular fruit type in south Asia. Lychees have high requirements for freshness preservation, so they are generally transported at low temperatures. Additionally, the surface of lychees is soft and subject to scratches and cracks during harvesting and processing. We build an image dataset with three classes of lychees, namely mature, defects, and rot. The dataset is annotated with ground truth bounding boxes for a supervised learning task. In addition, we train a transformer-based generative adversarial network (GAN) to generate synthetic samples that can effectively enrich the size and diversity of the original training set. We explore two design choices of DCNN-based object detection, namely one-stage vs. two-stage designs. Three DCNN models, including SSD-MobileNet V2, Faster RCNN-ResNet50,

and Faster RCNN-Inception-ResNet V2, are trained and evaluated on the self-created dataset. A set of extensive experiments show that GAN-based augmentation can help rebalance the training set with more diverse samples, which allows the DCNN models to capture better semantic features, resulting in notable performance gains. A performance comparison shows that SSD-MobileNet V2, trained on the augmented dataset, is on a par with the other two models in mean average precision (mAP) and significantly outperforms the other two in inference speed.

2. Materials and Methods

2.1. Dataset

2.1.1. Lychee Samples

The black leaf lychee samples were purchased in two batches from the Jiangbei fruit wholesale market in Huzhou city, Guangdong Province, China. The first batch had 2042 mature lychees, in which a total of 1216 samples were with cracks but not rotten. We placed 625 cracked lychee samples in a dry environment at room temperature for nature rot; the remaining 495 samples were exposed to sunlight until dried and rotten. The second batch consisted of 865 lychees that were used for the follow-up out-of-sample (OOS) testing. For both batches, the samples were placed in foam boxes with ice packs for freshness and shipped back directly to the laboratory once purchased.

2.1.2. Image Acquisition

To increase data diversity, lychee images were collected at six locations on the Huizhou College campus on 2 July 2019 (overcast) from 8:00 a.m. to 11:30 a.m. and from 5:00 p.m. to 7:30 p.m. The acquisition device was an iPhone 7 plus, the image format was JPEG, and the image resolution was 3024 by 3024 pixels. The camera was about 10–20 cm away from each lychee, and the images were acquired in different directions. A total of 5014 images of lychee were collected.

2.1.3. Dataset Creation

To reduce the computational load during training, we downscaled the images to 640 by 640 pixels. We manually labeled the dataset using LabelImg (available at https://github.com/tzutalin/labelImg accessed on 20 September 2020), which can generate an XML file with a 4-tuple parameter list specifying the four coordinate pairs of the bounding box of a labeled object. Figure 1 shows lychee samples of different categories labeled by labelImg.

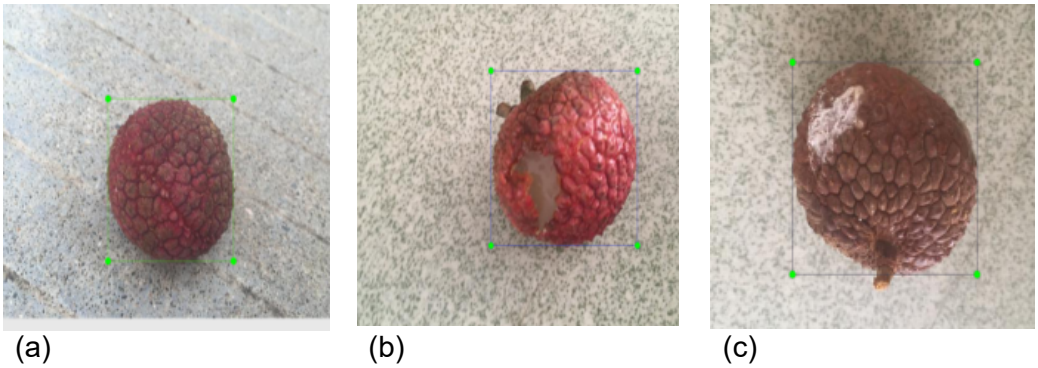


Figure 1. Lychee samples with annotated bounding boxes: (**a**) mature, (**b**) defects, and (**c**) rot.

2.1.4. Dataset Augmentation

Training a neural network needs many parameters, which generally require a large amount of labeled training data. To improve the generalization ability of the model and avoid overfitting, we augmented the original training set using GAN-based data augmentation. Before augmentation, the training set has a class imbalance problem,

where the three categories account for 44.03%, 25.75%, and 30.22%, respectively. The class imbalance problem usually leads to poor performance [34]. The GAN can be trained on the original training set and generate synthetic images for all categories, especially underrepresented ones. For our case, we obtain 1400 samples, including original and synthetic ones, for each category, as shown in the last column of Table 1.

Table 1. The lychee surface defect dataset.

Category	Original	Original %	Training	Test	Generated	Aug. Training
Mature	1648	44.03%	1298	350	102	1400
Defects	964	25.75%	800	164	600	1400
Rot	1331	30.22%	896	235	504	1400
Total	3743	100.00%	2994	749	1206	4200

2.2. The Proposed Learning Framework

Figure 2 describes the overall learning framework for lychee surface defect detection. The first and the second batch of lychee samples were used to create the original training and test sets, respectively. The original training set was fed into a GAN-based augmentation module to generate more training samples. The augmented training set was used to train the DCNN models, which were validated on the test set for a performance comparison.

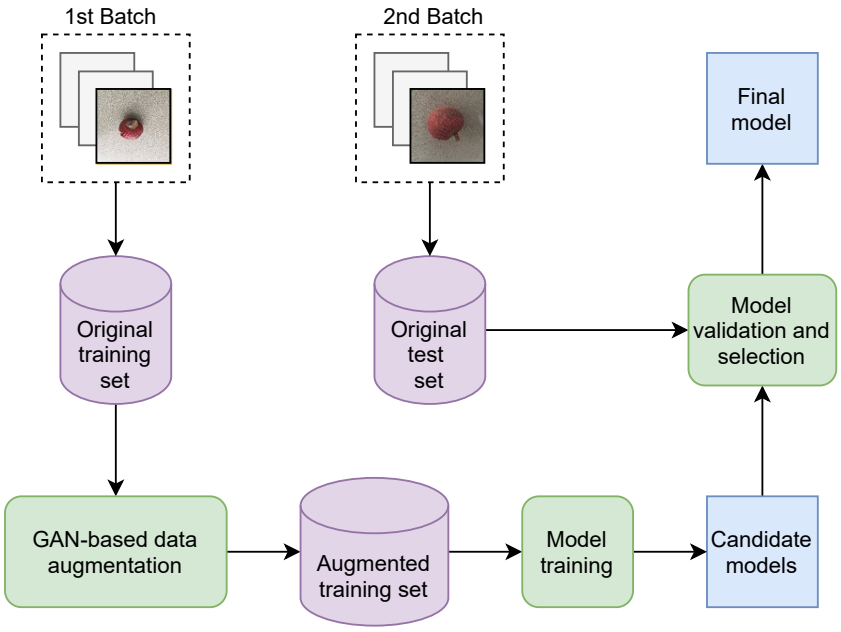


Figure 2. Learning Framework.

2.3. GAN-Based Data Augmentation

2.3.1. Training GAN

As a rising generative model, GAN can learn the distribution of a population of samples to generate realistic samples similar to the original ones. Due to the strong capability of generative modeling, GAN has been extensively used for data augmentation, especially in a low-resource scenario. The samples generated by GAN can be added to the training set to enhance the quantity and diversity of training data as well as boost the model performance. Figure 3 displays a diagram of the GAN utilized in this study. A typical GAN consists of two neural networks, a generator and a discriminator. The generator takes as input a random vector and aims to generate a fake sample that looks like a real one from the original dataset. On the other hand, the discriminator is trained with both real and fake samples and predicts whether or not a given sample is real. This prediction result is back-propagated through both networks to optimize the parameters.

Formally, let G and D denote the generator and the discriminator, respectively; let z denote the random vector; let $G(z)$ be the generated fake sample, which has the same dimension as a real sample x; let $\{(x_i, 1)\}_{i=1}^{m}$ and $\{(G(z_i), 0)\}_{i=1}^{m}$ be the sets of labeled real and fake samples, respectively, each with m samples that are utilized to jointly train the discriminator, which outputs a binary value $D(x)$, i.e., being real ($D(x) = 1$) or fake ($D(x) = 0$). During training, the discriminator aims to maximize the chance to distinguish real and fake samples. In other words, if an input sample x is real, D strives to push $D(x)$ towards 1, resulting in a positive prediction, and vice versa. The maximization problem for D is formulated in Equation (1). Similarly, the generator G aims to minimize the chance that $G(z)$ is detected by D as fake; the minimization problem for G is provided in Equation (2). Combining the two problems, we obtain a MinMax game, formulated in Equation (3). The goal of a GAN is to solve the MinMax game with an equilibrium, which describes a state when both G and D are sufficiently optimized and converge.

$$\max_{D} V(D) = \mathbb{E}_{x \sim P_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim noise}[\log(1 - D(G(z)))] \tag{1}$$

$$\min_{G} V(G) = \mathbb{E}_{z \sim noise}[\log(1 - D(G(z)))] \tag{2}$$

$$\min_{G} \max_{D} V(G, D) = \mathbb{E}_{x \sim P_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim noise}[\log(1 - D(G(z)))] \tag{3}$$

The training process is briefly described as follows. The training undergoes multiple epochs, each of which is divided into multiple time steps. For each time step, we first sample a batch of random vectors $\{z_i\}_{i=1}^{m}$ from the noise prior and sample a batch of real samples $\{x_i\}_{i=1}^{m}$ from the training set; we then update D by ascending its stochastic gradient $\Delta_{\Theta_d} \frac{1}{m} \sum_{i=1}^{m} [\log D(x_i) + \log(1 - D(G(z_i)))]$. After k time steps, we sample another batch of random vectors $\{z_i\}_{i=1}^{m}$, and update G by descending its stochastic gradient $\Delta_{\Theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(G(z_i)))$. This way, D and G are optimized alternatively until convergence. Once the GAN is trained, the generator can produce samples that are similar to the real ones in our dataset. We can then create new training samples to obtain an augmented training set. The efficacy of the GAN-based data augmentation is validated in Section 3.

2.3.2. TransGAN

TransGAN [35], is an effort to build a GAN based on pure Transformers, without the use of convolutions. A transformer encoder [36] consists of a multi-head self-attention module stacked by a feed-forward multilayer perceptron (MLP) to capture long-term dependency and contextual semantic information among words in a sentence. Although created for NLP applications, transformer has been adopted in computer vision [37], where an image is partitioned into multiple patches. In TransGAN, both the generator and discriminator are built with transformer encoder blocks, as shown in Figure 3. To reduce memory consumption resulting from a large amount of patches, TransGAN adopts a multi-stage mechanism that progressively upscale/downscale the image resolution. In addition, a grid self-attention module is developed to first partition a full feature map into non-overlapping grids and then replace the global token-wise attention with a local attention within a grid, greatly reducing the computation load. Both design tricks have been validated in [35] and demonstrated superior performances in generative modeling, compared with the state of the art, in several well-known datasets. Therefore, we adopted TransGAN as a building block of the proposed learning framework.

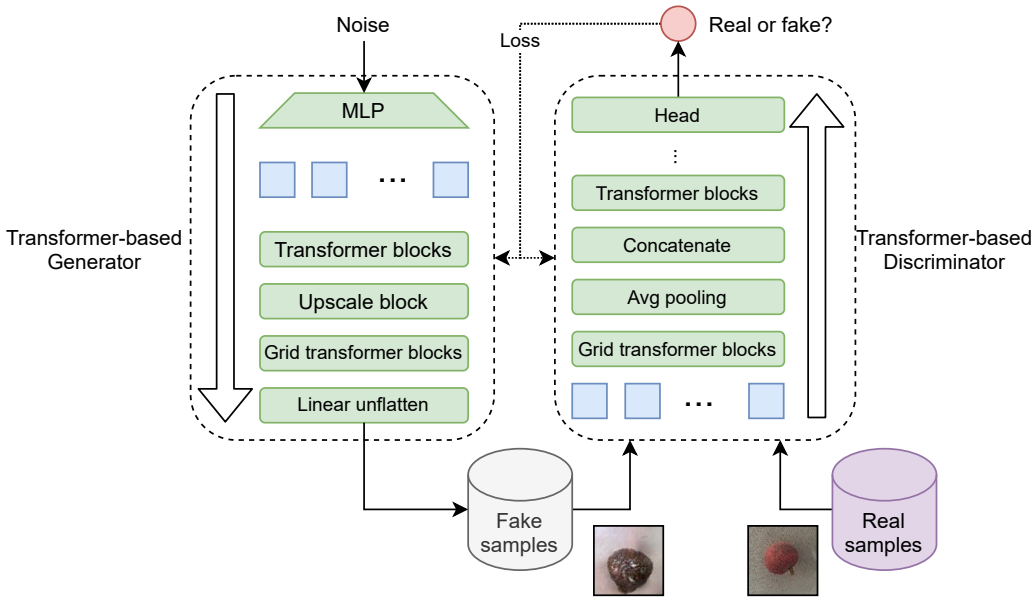


Figure 3. TransGAN architecture.

2.4. DCNN Models

This section gives a brief description of the three DCNN-based object detection models, including SSD-MobileNet V2, Faster RCNN-ResNet50, and Faster RCNN-Inception-ResNet V2.

2.4.1. Model 1: SSD-MobileNet V2

SSD makes it possible to perform object localization and classification in a single forward pass of the network. SSD adopts the VGG-16 architecture as a backbone network and adds a set of auxiliary convolutional layers that allow feature extraction at multiple scales and progressively decrease the size of the input to each subsequent layer. SSD uses multi-scale feature mapping, that is, to simulate the process of seeing things in the human eye. The algorithm uses the combination of multi-level feature maps as the basis of classification and regression. The purpose of this is to accurately detect object features on various scales. In the low-level feature map, the receptive field is relatively small, and the high-level receptive field is rather large. In SSD, default boxes with different sizes and aspect ratios are used for feature maps of different scales. At the same time, SSD adopts the default box mechanism and takes several different aspect ratios for the feature units on the same feature layer.

MobileNet [38] is a lightweight and efficient deep neural network. MobileNet v1 consists of an input layer, 13 convolution layers, an average pooling layer, and a fully connected layer. Additionally, a Batch Normalization (BN) layer and a ReLU activation are added after each convolution layer. Two hyper-parameters, a width multiplier and resolution multiplier, were introduced to allow model developers to trade off latency or accuracy for speed and small size depending on the application requirements. The former reduces the number of feature maps to control the thickness of the model, and the latter reduces the size of feature maps to alleviate the amount of computation. MobileNet uses a deep separable convolution (Depthwise Separable Convolution) to decompose and calculate the standard convolution kernel, which reduces the amount of calculation. Depthwise separable convolution divides a standard convolution into a depthwise convolution and a 1×1 convolution (pointwise convolution). Each input channel is spatially convolved separately by the depth-wise convolution; then, the resulting outputs are mixed via pointwise convolutions.

On top of v1, two new features are introduced in MobileNet V2 [39], including a linear bottleneck between the layers and a shortcut connection between the bottlenecks. A linear bottleneck aims to encode intermediate inputs and outputs that facilitate the inner layers to

learn transforming pixel-level concepts to higher-level descriptors. Additionally, a shortcut connection enables more efficient training for a deep network.

Replacing the VGG-16 backbone of SSD with the MobileNet V2, we obtain SSD-MobileNet V2, which combines the pros of SSD and MobileNet V2 models to reduce the computational load. Figure 4 shows the neural architecture of SSD-MobileNet V2, which employs MobileNet V2 as a feature extractor, followed by a set of extra layers for multi-scale learning.

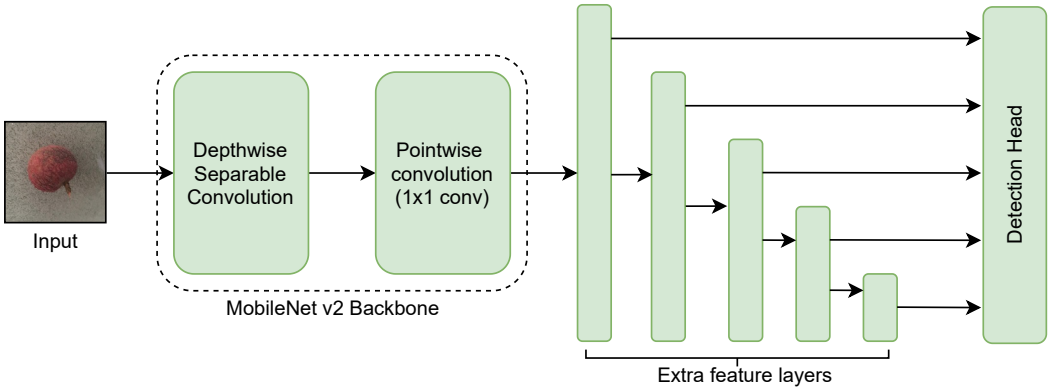


Figure 4. Neural architecture of SSD-MobileNet V2.

2.4.2. Model 2: Faster RCNN-ResNet50

Girshick et al. [40] proposed Regions with CNN features (RCNN) that performs a selective search to extract 2000 regions from an image, called region proposals. Each candidate region proposal is warped into a square and fed into a CNN that generates a 4096-dimensional feature vector as output, which is passed to a support vector machine (SVM) for classification. RCNN started a new direction for object detection. However, it took about 47 s to process one image, which was not suitable for real-time detection.

To speed up RCNN, the same research group proposed Fast RCNN [7]. Rather than feeding the region proposals to the CNN, Fast RCNN feeds the input image to the CNN to produce a convolutional feature map, from which region of proposals are identified and warped into squares to form a feature vector. Fast RCNN is faster because the CNN does not process 2000 region proposals every time. Instead, the convolution operation is carried out only once per image and a feature map is generated from it.

Both RCNN and Fast RCNN employ a selective search algorithm to propose regions, which is time-consuming. Ren et al. [6] propose Faster RCNN, which eliminates the selective search and lets the network learn region proposals. Faster RCNN introduces a region proposal network (RPN) that takes an image (of any size) as input and outputs a set of rectangular object proposals, each with an objectness score and uses non-maximal suppression for region scores. The RPN outputs its TOP-N scored region suggestions to the region of interest (ROI) pooling layer, which generates a fixed-size proposal feature map, then feeds the map into the subsequent classification regression network through a fully connected layer. The classification layer predicts the category of a proposed frame and performs the bounding box regression to obtain the final accurate location of the detected object, as shown in Figure 5.

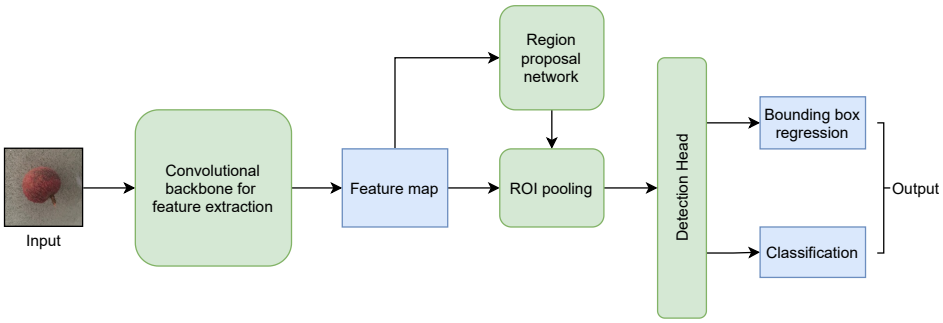


Figure 5. Faster RCNN.

Simply stacking layers together to deepen a neural network suffers the notorious vanishing gradient problem—namely, as gradients are back-propagated to earlier layers, repeated multiplication may make the gradient infinitely small, leading to rapid performance degradation. ResNet [41] allows training a neural network with up to hundreds or even thousands of layers and still achieves compelling performance. ResNet introduces an identity shortcut connection that skips one or more layers. The network transforms a residual mapping function $F(x) = H(x) - x$ via an identity mapping function $H(x) = F(x) + x$ (H (x) is an ideal mapping, which is the final mapping output). If $F(x) = 0$, it forms an identity mapping $H(x) = x$, and it is easier to fit the residual. Figure 6a shows a typical residual block with a shortcut (or skip connection).

ResNet50 is a 50-layer residual network, which consists of two basic structures, an identity block and a convolutional block. The main difference between these two is whether the convolution operation is performed on the shortcut connection. The use of Faster RCNN combined with a ResNet50 network can deepen the neural network as much as possible while maintaining the optimal state of a DCNN during training.

2.4.3. Model 3: Faster RCNN-Inception-ResNet V2

An Inception module [42] offers superior local topology for a neural network. Specifically, the module, as shown in Figure 6b, performs multiple convolutional or pooling operations on the input image in parallel and stitches all the results into a deep feature map. It uses different filters to perform convolution operations on the input to obtain different information about the input image. Processing these operations in parallel and combining all the feature maps would result in a better image representation. Common versions of Inception networks are Inception V1, Inception V2, Inception V3, Inception V4, and Inception ResNet V1/V2.

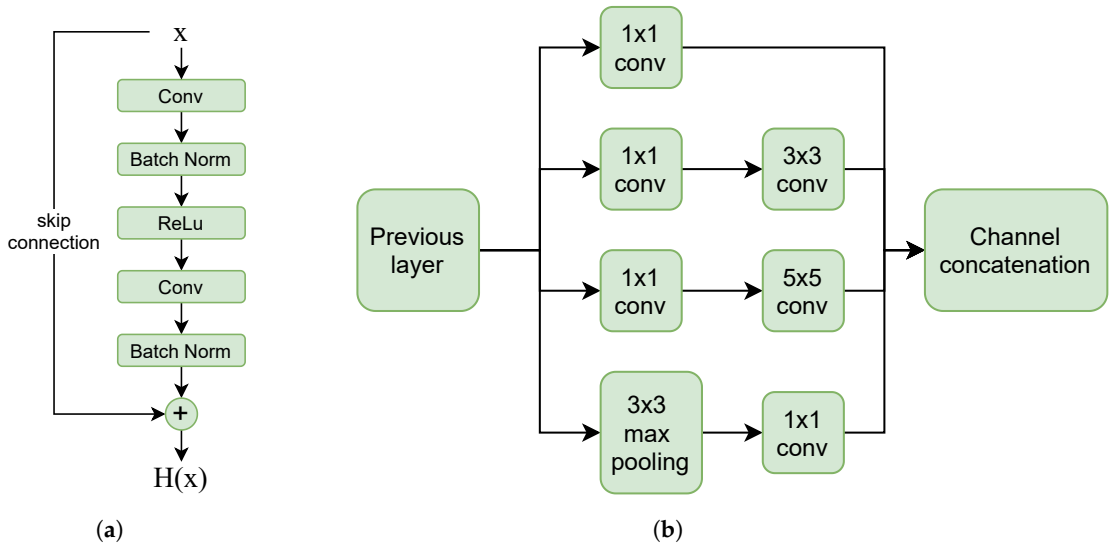


Figure 6. (**a**) Res block; (**b**) inception module.

The Inception-ResNet V2 model [43] incorporates numerous ResNet blocks into the Inception V4 network to allow training a deeper neural network.

3. Results and Discussion

3.1. Experimental Platform

The experiments were carried out on a computer with a 16 GB RAM, an Intel Core i7-8700 CPU, and a Windows 10 operating system. To train a DCNN model, we employed the TensorFlow framework, along with an Nvidia GeForce RTX 2070 GPU. The program was written in Python 3.6.7.

3.2. Training Setting

To train TransGAN, we followed the settings in [35]. We adopted a learning rate of 0.0001, an Adam optimizer, a batch size of 64, for both generator and discriminator. The TransGAN was trained with 220 epochs.

We trained the SSD-MobileNet V2, Faster RCNN-ResNet50, and Faster RCNN-Inception-ResNet V2 models on the original and augmented training set shown in Table 1, respectively, generating six models for comparison. Each trained model takes an image as input and can output a bounding box for each detected lychee object in the image with a predicted category and a confidence score. For the hyper-parameter setting, we used a weight decay of 0.0005, a momentum of 0.8, a verification period of 5000, a batch size of 32, a learning rate of 0.005, and performed 1500 epochs of training.

3.3. Generated Samples

The trained TransGAN model does not always generate high-quality image samples. For the mature/defects/rot categories, the ratios of kept and total generated samples are 102/200, 600/1000, and 504/800. Figure 7 shows a set of lychee samples generated by the trained TransGAN. For each category, we select two samples with good and moderate quality. The samples with good quality, Figure 7a–c, present realistic and clear texture features appearing in the real samples. For examples, the dark red and uncracked surface for the mature ones, the exposing sarcocarp in the crack for the defective ones, and the mildew spots on the rot ones have been well generated. Compared to Figure 7d–f, the samples Figure 7a–c present more realistic looking in texture, shape, color, and size, which are the criteria used to determine which generated samples are added to the training set.

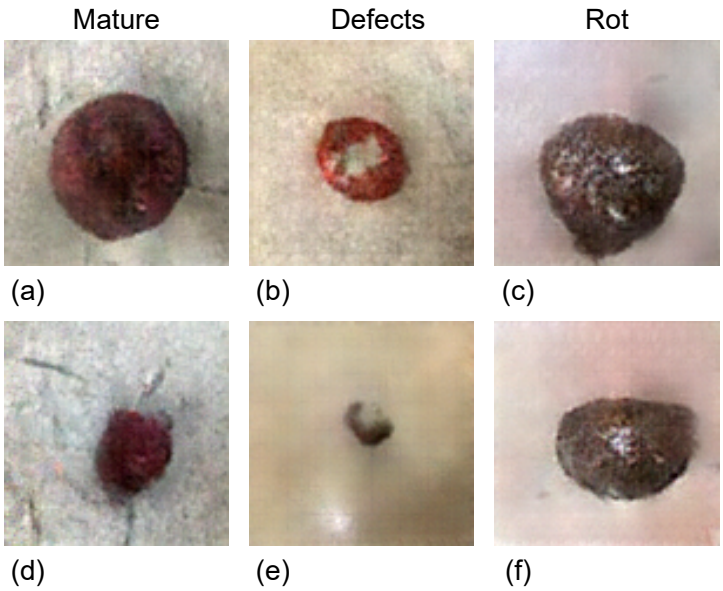


Figure 7. Samples generated by GAN. (**a**–**c**) are three generate samples with good quality, and (**d**–**f**) are three generated ones with poor quality.

3.4. Evaluation Metrics

We employed the mean average precision (mAP) and the detection speed as the two performance metrics, which are commonly used in object detection. The mAP is calculated based on Intersection over Union (IoU), which is defined by the ratio of the area of intersection and area of union of the predicted and ground truth bounding boxes. Typically, a true positive (TP) is identified if IoU > 0.5; otherwise, a false positive (FP) is counted. Additionally, a false negative (FN) is counted if (1) there is no detection at all or (2) the detection has an IoU > 0.5 but misclassifies an object. Precision (Pre) is given as the ratio of true positive (TP) and the total number of predicted positives. Recall (Rec) is defined as the ratio of TP and total of ground truth positives. The interpolated precision is calculated at each recall level by taking the maximum precision measured for that level. The average precision (AP) is then calculated by taking the area under the precision-recall curve. The mAP is the average of the AP calculated for all the classes across all images in the test set. In general, a higher mAP indicates a better model. In addition, we measured the inference speed using frames per second to evaluate whether a model can satisfy the requirement of real-time detection.

In addition to mAP, which is usually utilized in object detection, we also considered four metrics widely adopted in classification, including Accuracy (Acc), Recall (Rec), Specificity (Spe), and F1-score (F1). These four metrics are defined as follows.

$$\text{Acc} = \frac{\text{TP} + \text{FN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{4}$$

$$\text{Rec} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{5}$$

$$\text{Spe} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{6}$$

$$\text{F1} = 2 \times \frac{\text{Pre} \times \text{Rec}}{\text{Pre} + \text{Rec}} \tag{7}$$

3.5. Overall Performance

Figure 8 show a performance comparison of the three models trained with and without GAN-based augmentation on the test set. We provide the observations as follows.

- It is observed that before augmentation, the mAPs of SSD-MobileNet V2, Faster RCNN-ResNet50, and Faster RCNN-Inception-ResNet V2 were 88.95%, 91.57%, and 91.25%, respectively. Training on the GAN-augmented data, the performance gains are 2.86%, 1%, and 0.58% for SSD-MobileNet V2, Faster RCNN-ResNet50, and RCNN-Inception-ResNet V2, respectively. SSD-MobileNet V2 underwent the largest gain, becoming comparable to the other two models.
- Before augmentation, the performance gap between individual classes is large, caused by imbalanced sample distribution in the original dataset. For all three models, the mature and the defects classes received the highest and lowest mAP, presenting a gap of 9.45%, 6.12%, and 7.77% for SSD-MobileNet V2, Faster RCNN-ResNet50, and Faster RCNN-Inception-ResNet V2, respectively. After augmentation, the training set was rebalanced, and the corresponding gaps were reduced to 1.78%, 4.45%, 2.35%, respectively, indicating that the models are trained to optimize the mAP for three classes more equally. The resulting models are therefore more practical because more defective and rotten samples can be detected.
- Faster RCNN-ResNet50 shows the best overall mAP among the three models, with the highest mAP for two individual classes, namely, mature and defects. Faster RCNN-Inception-ResNet V2, on the other hand, performs the best on rotten samples.

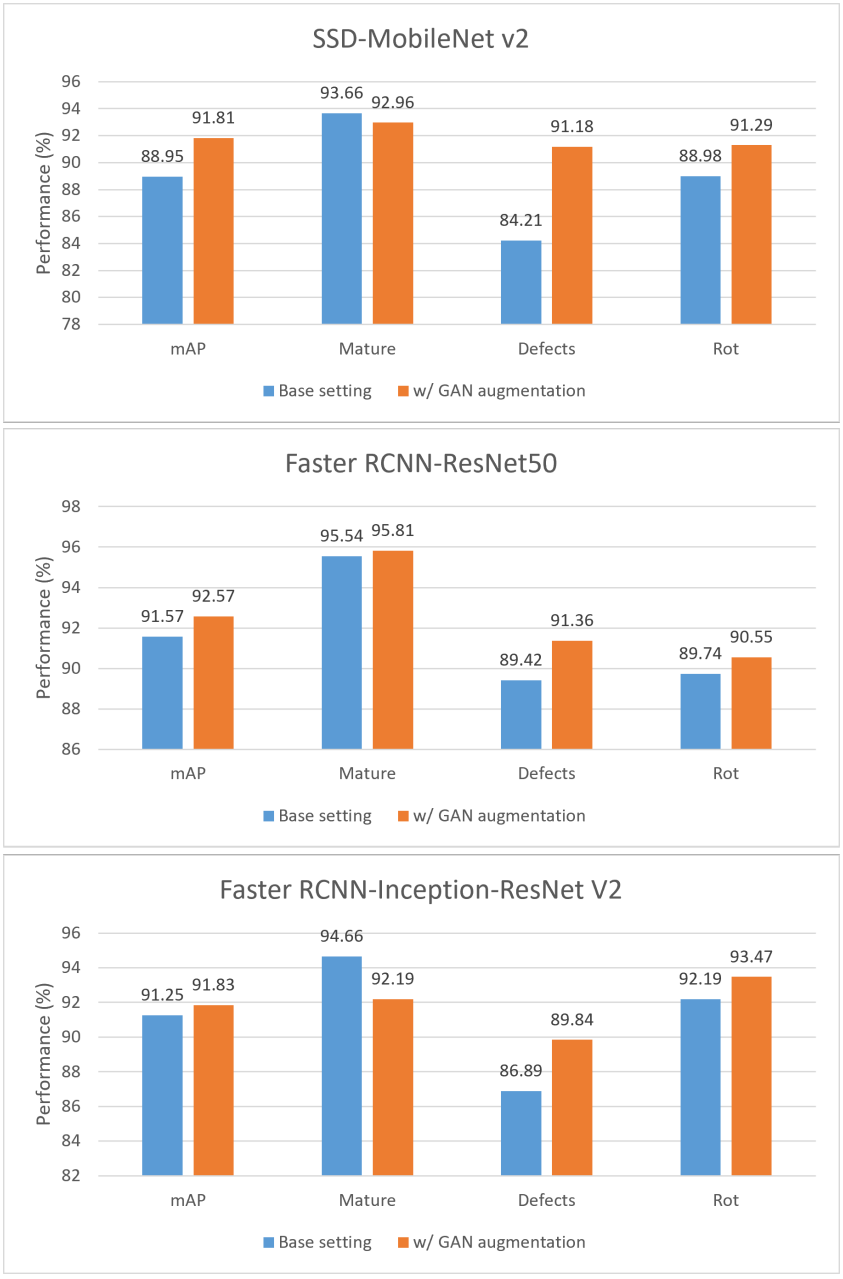


Figure 8. MAP comparison of the three models for the three lychee categories.

Figure 9 shows the inference speed for the three models. We observed that all three meet the real-time detection requirement, with an average detection speed of 102 FPS, 68 FPS, and 23 FPS for SSD-MobileNet V2, Faster RCNN-ResNet50, and Faster RCNN-Inception-ResNet V2, respectively. Being 1.5 times faster than Faster RCNN-ResNet50 and 4.4 times faster than RCNN-Inception-ResNet V2, SSD-MobileNet V2 presents a superior inference speed owing to its lightweight architecture, which was originally designed for mobile devices. The efficiency advantage brought by SSD-MobileNet V2 is mainly due to its nature of being a one-stage object detection method, and the Faster RCNN family operates in two stages, including RoI proposal/selection and classification, which adds computational load to the network. SSD, on the other hand, was trained to run detection directly over a dense sampling of anchors without the stage of RoI selection.

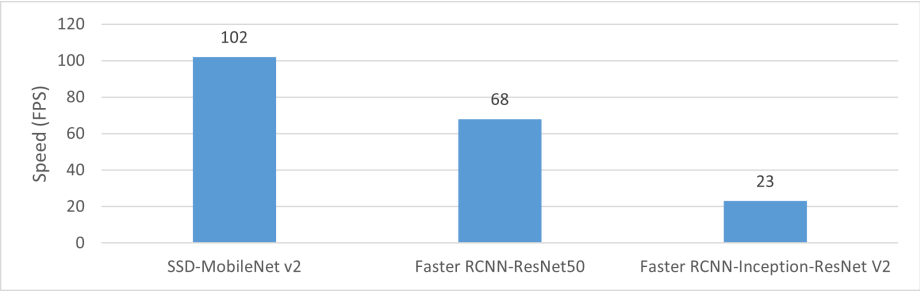


Figure 9. Comparison of the three models in detection speed.

In our task, SSD-MobileNet V2 was worse than both Faster RCNN-ResNet50 and Faster RCNN-Inception-ResNet V2 by 3%–4%. However, with a more balanced and diverse training set augmented by TransGAN, SSD-MobileNet V2 was able to achieve a comparable mAP, with a significant gain in the defects and rot categories.

From a classification perspective, we report the confusion matrices in Figure 10 and derive the model performance in Acc, Rec, Spe, and F1 in Table 2 for the three investigated models before and after augmentation. It was observed that all three models present consistent performance gains with the GAN-based augmentation method applied. Similar to the mAP results, SSD-MobileNet V2 had the largest gains of around 2% in all four metrics (shown in Table 2), making it comparable with the other two. Combining the results of all models in mAP, Acc, Rec, Spe, F1, and inference speed, we conclude that SSD-MobileNet V2 is the most cost-effective model. Faster RCNN-ResNet50 is the second-best overall, with the highest detection and classification performance and the second highest speed.

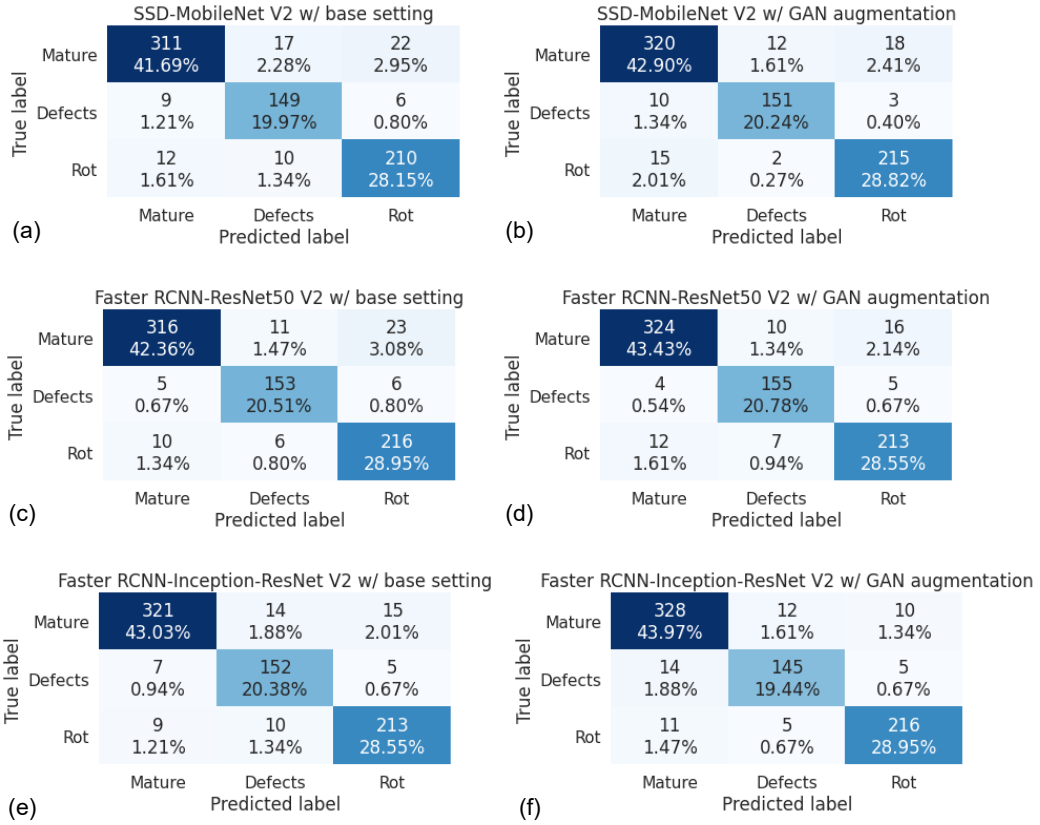


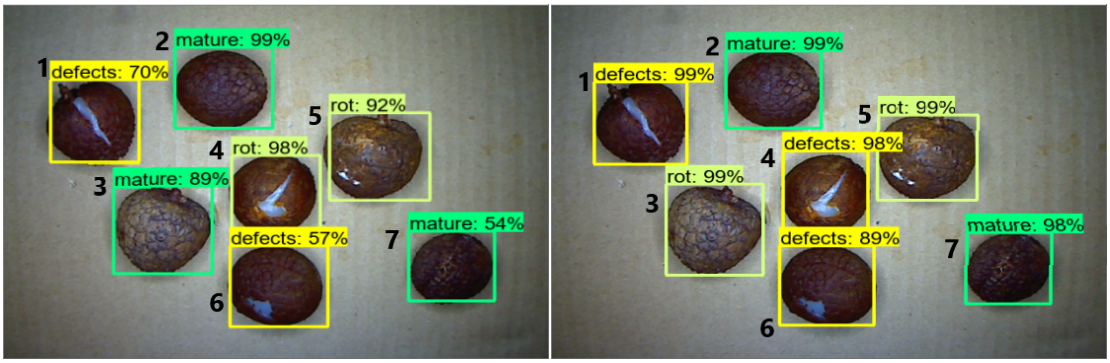
Figure 10. Confusion matrices. (**a**) SSD-MobileNet V2 w/ base setting. (**b**) SSD-MobileNet V2 w/ GAN augmentation. (**c**) Faster RCNN-ResNet50 w/ base setting. (**d**) Faster RCNN-ResNet50 w/ GAN augmentation. (**e**) Faster RCNN-Inception-ResNet V2 w/ base setting. (**f**) Faster RCNN-Inception-ResNet V2 w/ GAN augmentation.

Table 2. Classification performance. Abbreviations: SSD-MobileNet V2 (S.M. V2); Faster RCNN-ResNet50 (F.R.R50); Faster RCNN-Inception-ResNet V2 (F.R.I.R. V2).

		Acc	Rec	Spe	F1
S.M. V2	w/base setting	89.81%	90.08%	89.89%	89.46%
	w/GAN augmentation	91.96%	92.06%	91.99%	91.92%
F.R.R50	w/base setting	91.82%	92.23%	91.95%	91.72%
	w/GAN augmentation	92.76%	92.96%	92.80%	92.55%
F.R.I.R. V2	w/base setting	91.96%	92.07%	91.98%	91.54%
	w/GAN augmentation	92.36%	91.74%	92.22%	91.86%

3.6. Qualitative Results

To obtain an impression on the detection effect before and after augmentation, we took an image with seven samples hand-picked from the test set, including two mature ones (samples 2 and 7), three with defects (samples 1, 4, and 6), and two rotten ones (samples 3 and 5). We applied the SSD-MobileNet V2 model, trained with and without augmentation, to the image. The detection results are in Figure 11. It was observed that, with the base setting (i.e., no augmentation), SSD-MobileNet V2 misclassified a rotten lychee (sample 3) as mature and a defective one (sample 4) as rotten; additionally, the confidence scores for samples 1, 6, and 7 were low, indicating the model's weak ability to distinguish lychees of different categories. After GAN-based augmentation, SSD-MobileNet V2 presented notable improvements in both bounding box accuracy (samples 2 and 5) and confidence score (as seen in samples 1, 5, 7, and 6); most importantly, the two classification errors (3 and 4) were fixed.



(**a**) (**b**)

Figure 11. Detection by (**a**) SSD-MobileNet V2 w/base setting and (**b**) SSD-MobileNet V2 w/GAN-based augmentation.

To further investigate the effect of GAN-based augmentation on feature extraction, we hand-picked three samples, with one from each class, and tested SSD-MobileNet V2 and Faster RCNN-ResNet50 on the three samples. To visualize the learned features, we plotted the feature map activation via a heatmap, as shown in Figure 12, where the critical regions with rich pattern information are highlighted. The figure contains an image matrix, where the three rows represent the three classes, and the five columns consist of the original image set (column 1) and the images with activation heatmaps, generated by the two models trained under two settings (columns 2–5). We list some notable findings as follows.

- For SSD-MobileNet V2, we compared columns two and three and found that for the mature sample (images b and c), fewer regions on the sample surface were activated after augmentation, indicating a potential confidence drop; for the defective sample (images g and h), a stronger activation spot appears at the left side of the crack on the surface, shown in image h, which could result in confidence gain; lastly, for the rotten sample (images i and m), there is no apparent activation change observed.

- For Faster RCNN-ResNet50, we compared columns four and five and observe the following activation differences: for the mature sample (images d and e), more regions of the sample surface have been activated with the augmented setting, but the signals are weaker, which could lead to mixed effect; a similar observation can be found for the defective sample (from image i to j) and the rotten sample (from image n to o). Additionally, another important finding is that for all three samples, the activation regions have expanded and are more centralized, which would result in more accurate bounding box localization.
- For both SSD-MobileNet V2 and Faster RCNN-ResNet50 under the augmented setting, the activation contours of all samples have shrunken and fit better to the sample boundary, meaning that the models have learned to improve the bounding box regression, which yields a better IoU score.

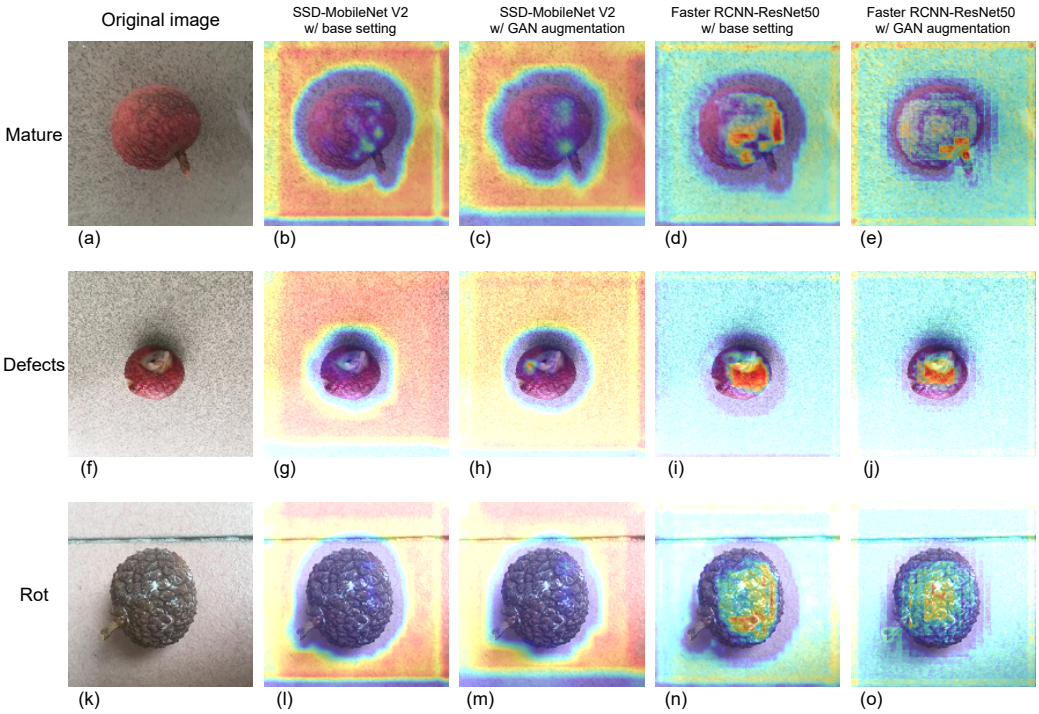


Figure 12. Activation heatmap. (**a**,**f**,**k**) are the original samples; (**b**,**g**,**l**) are the heatmaps for SSD-MobileNet V2 w/ base setting; (**c**,**h**,**m**) are the heatmaps for SSD-MobileNet V2 w/ GAN augmentation; (**d**,**i**,**n**) are the heatmaps for Faster RCNN-ResNet50 w/ base setting; (**e**,**j**,**o**) are the heatmaps for Faster RCNN-ResNet50 w/ GAN augmentation.

4. Conclusions

The surface of lychees is subject to scratches and cracks during harvesting/processing. Additionally, lychees are hard to preserve and have to be stored in low temperatures to keep fresh. To explore the automation of defective surface detection for lychees, we have built an image dataset with three classes of lychees, namely mature, defects, and rot. In addition, we trained a transformer-based GAN to generate synthetic samples that can effectively enrich the size and diversity of the original training set. Three DCNN models, including SSD-MobileNet V2, Faster RCNN-ResNet50, and Faster RCNN-Inception-ResNet V2, were trained and evaluated on the self-created dataset. A set of extensive experiments show that GAN-based augmentation can help rebalance the training set with more diverse samples, which allows the DCNN models to capture better semantic features, resulting in notable performance gains. The results show that SSD-MobileNet V2, trained on the augmented dataset, presents a mAP of 91.81% with an inference speed of 102 FPS, making it the most cost-effective model among the three.

This study has the following limitations that will be addressed in future work. First, GAN-based data augmentation can be combined with classic image processing-based augmentation to quickly enhance the dataset in both quantity and diversity. It would be interesting to explore the individual and joint effects of these two types of augmentations and how well they can complement each other to maximize the benefit of augmentation. Second, this study only investigated one possible GAN architecture due to its validated performance to generate high-quality synthetic images. At the same time, numerous GAN options that can also be evaluated. Third, samples created by GAN-based augmentation require manual selection and annotation before they can be used for training, which is time-consuming. It would be worthwhile developing additional supporting algorithms for automated image quality scanning and bounding box annotation.

Author Contributions: Conceptualization and methodology, C.W. and Z.X.; software, validation, and original draft preparation, C.W.; review and editing, supervision, Z.X. Both authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: The data used in this study are available on request from the authors.

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

References

1. Razmjooy, N.; Mousavi, B.S.; Soleymani, F. A real-time mathematical computer method for potato inspection using machine vision. *Comput. Math. Appl.* **2012**, *63*, 268–279. [CrossRef]
2. Zhou, Z.; Huang, Y.; Li, X.; Wen, D.; Wang, C.; Tao, H. Automatic detecting and grading method of potatoes based on machine vision. *Trans. Chin. Soc. Agric. Eng.* **2012**, *28*, 178–183.
3. Wang, C.; Li, X.; Wu, Z.; Zhou, Z.; Feng, Y. Machine vision detecting potato mechanical damage based on manifold learning algorithm. *Trans. Chin. Soc. Agric. Eng.* **2014**, *30*, 245–252.
4. Yao, L.; Lu, L.; Zheng, R. Study on Detection Method of External Defects of Potato Image in Visible Light Environment. In Proceedings of the 2017 10th International Conference on Intelligent Computation Technology and Automation (ICICTA), Changsha, China, 9–10 October 2017; pp. 118–122.
5. Xie, W.; Wang, F.; Yang, D. Research on carrot surface defect detection methods based on machine vision. *IFAC-PapersOnLine* **2019**, *52*, 24–29. [CrossRef]
6. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *39*, 1137–1149. [CrossRef]
7. Girshick, R. Fast r-cnn. In Proceedings of the 2015 IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1440–1448
8. Litjens, G.; Kooi, T.; Bejnordi, B.E.; Setio, A.A.A.; Ciompi, F.; Ghafoorian, M.; Van Der Laak, J.A.; Van Ginneken, B.; Sánchez, C.I. A survey on deep learning in medical image analysis. *Med. Image Anal.* **2017**, *42*, 60–88. [CrossRef]
9. Minaee, S.; Boykov, Y.Y.; Porikli, F.; Plaza, A.J.; Kehtarnavaz, N.; Terzopoulos, D. Image segmentation using deep learning: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**. [CrossRef]
10. Chen, Y.; Tian, Y.; He, M. Monocular human pose estimation: A survey of deep learning-based methods. *Comput. Vis. Image Underst.* **2020**, *192*, 102897. [CrossRef]
11. Grigorescu, S.; Trasnea, B.; Cocias, T.; Macesanu, G. A survey of deep learning techniques for autonomous driving. *J. Field Robot.* **2020**, *37*, 362–386. [CrossRef]
12. Guo, Y.; Liu, Y.; Oerlemans, A.; Lao, S.; Wu, S.; Lew, M.S. Deep learning for visual understanding: A review. *Neurocomputing* **2016**, *187*, 27–48. [CrossRef]
13. He, Y.; Song, K.; Meng, Q.; Yan, Y. An end-to-end steel surface defect detection approach via fusing multiple hierarchical features. *IEEE Trans. Instrum. Meas.* **2019**, *69*, 1493–1504. [CrossRef]
14. Tao, X.; Zhang, D.; Ma, W.; Liu, X.; Xu, D. Automatic metallic surface defect detection and recognition with convolutional neural networks. *Appl. Sci.* **2018**, *8*, 1575. [CrossRef]
15. Siddiqi, R. Automated apple defect detection using state-of-the-art object detection techniques. *SN Appl. Sci.* **2019**, *1*, 1–12. [CrossRef]
16. Kayaalp K.; Metlek, S. Classification of robust and rotten apples by deep learning algorithm. *Sak. Univ. J. Comput. Inf. Sci.* **2020**, *3*, 112–120. [CrossRef]

17. Alam, M.N.; Saugat, S.; Santosh, D.; Sarkar, M.I.; Al-Absi, A.A. Apple Defect Detection Based on Deep Convolutional Neural Network. In *International Conference on Smart Computing and Cyber Security: Strategic Foresight, Security Challenges and Innovation*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 215–223.
18. Zhu, H.; Yang, L.; Sun, Y.; Han, Z. Identifying carrot appearance quality by an improved dense CapNet. *J. Food Process. Eng.* **2021**, *44*, e13586. [CrossRef]
19. Xie, W.; Wei, S.; Zheng, Z.; Jiang, Y.; Yang, D. Recognition of Defective Carrots Based on Deep Learning and Transfer Learning. *Food Bioprocess Technol.* **2021**, *14*, 1361–1374. [CrossRef]
20. Azizah, L.M.; Umayah, S.F.; Riyadi, S.; Damarjati, C.; Utama, N.A. Deep learning implementation using convolutional neural network in mangosteen surface defect detection. In Proceedings of the 2017 7th IEEE International Conference on Control System, Computing and Engineering (ICCSCE), Penang, Malaysia, 24–26 November 2017; pp. 242–246.
21. da Costa, A.Z.; Figueroa, H.E.; Fracarolli, J.A. Computer vision based detection of external defects on tomatoes using deep learning. *Biosyst. Eng.* **2020**, *190*, 131–144. [CrossRef]
22. Zhou, H.; Zhuang, Z.; Liu, Y.; Liu, Y.; Zhang, X. Defect Classification of Green Plums Based on Deep Learning. *Sensors* **2020**, *20*, 6993. [CrossRef]
23. Tian, Y.; Yang, G.; Wang, Z.; Li, E.; Liang, Z. Detection of apple lesions in orchards based on deep learning methods of cyclegan and yolov3-dense. *J. Sens.* **2019**, *2019*, 7630926. [CrossRef]
24. Goodfellow, I.J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial networks. *arXiv* **2014**, arXiv:1406.2661.
25. Choi, J.; Kim, T.; Kim, C. Self-ensembling with gan-based data augmentation for domain adaptation in semantic segmentation. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 27 October– 2 November 2019; pp. 6830–6840.
26. Huang, S.W.; Lin, C.T.; Chen, S.P.; Wu, Y.Y.; Hsu, P.H.; Lai, S.H. AugGAN: Cross Domain Adaptation with GAN-based Data Augmentation. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018.
27. Rashid, H.; Tanveer, M.A.; Khan, H.A. Skin lesion classification using GAN based data augmentation. In Proceedings of the 2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Berlin, Germany, 23–27 July 2019; pp. 916–919.
28. Frid-Adar, M.; Diamant, I.; Klang, E.; Amitai, M.; Goldberger, J.; Greenspan, H. GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification. *Neurocomputing* **2018**, *321*, 321–331. [CrossRef]
29. Mahmood, R.; Babier, A.; McNiven, A.; Diamant, A.; Chan, T.C. Automated treatment planning in radiation therapy using generative adversarial networks. In Proceedings of the Machine Learning for Healthcare Conference, Palo Alto, CA, USA, 17–18 August 2018; pp. 484–499.
30. Dar, S.U.; Yurt, M.; Karacan, L.; Erdem, A.; Erdem, E.; Çukur, T. Image synthesis in multi-contrast MRI with conditional generative adversarial networks. *IEEE Trans. Med Imaging* **2019**, *38*, 2375–2388. [CrossRef] [PubMed]
31. Han, C.; Hayashi, H.; Rundo, L.; Araki, R.; Shimoda, W.; Muramatsu, S.; Furukawa, Y.; Mauri, G.; Nakayama, H. GAN-based synthetic brain MR image generation. In Proceedings of the 2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018), Washington, DC, USA, 4–7 April 2018; pp. 734–738.
32. Yi, X.; Walia, E.; Babyn, P. Generative adversarial network in medical imaging: A review. *Med. Image Anal.* **2019**, *58*, 101552. [CrossRef] [PubMed]
33. Kusiak, A. Convolutional and generative adversarial neural networks in manufacturing. *Int. J. Prod. Res.* **2020**, *58*, 1594–1604. [CrossRef]
34. Longadge, R.; Dongre, S. Class imbalance problem in data mining review. *arXiv* **2013**, arXiv:1305.1707.
35. Jiang, Y.; Chang, S.; Wang, Z. Transgan: Two transformers can make one strong gan. *arXiv* **2021**, arXiv:2102.07074.
36. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is all you need. *arXiv* **2017**, arXiv:1706.03762.
37. Chen, J.; Lu, Y.; Yu, Q.; Luo, X.; Adeli, E.; Wang, Y.; Lu, L.; Yuille, A.L.; Zhou, Y. Transunet: Transformers make strong encoders for medical image segmentation. *arXiv* **2021**, arXiv:2102.04306.
38. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
39. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Istanbul, Turkey, 30–31 January 2018; pp. 4510–4520.
40. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, Ohio, 24–27 June 2014; pp. 580–587.
41. 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, 26 June–1 July 2016; pp. 770–778.

42. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 8–10 June 2015; pp. 1–9.
43. Szegedy, C.; Ioffe, S.; Vanhoucke, V.; Alemi, A. Inception-v4, inception-resnet and the impact of residual connections on learning. In Proceedings of the AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; Volume 31.