Article

Detecting and Measuring Defects in Wafer Die Using GAN and YOLOv3

Ssu-Han Chen ^{1,2,*} , Chih-Hsiang Kang ^{1,2} and Der-Baau Perng ^{3,4}¹ Department of Industrial Engineering and Management, Ming Chi University of Technology, New Taipei City 24301, Taiwan; C.H.Kang@o365.mcut.edu.tw² Center for Artificial Intelligence & Data Science, Ming Chi University of Technology, New Taipei City 24301, Taiwan³ Department of M-Commerce and Multimedia Applications, Asia University, Taichung 41354, Taiwan; perng@asia.edu.tw⁴ Department of Business Management, Asia University, Taichung 41354, Taiwan

* Correspondence: ssuhanchen@mail.mcut.edu.tw

Received: 16 October 2020; Accepted: 3 December 2020; Published: 5 December 2020

Featured Application: Die defect detection and measurement.

Abstract: This research used deep learning methods to develop a set of algorithms to detect die particle defects. Generative adversarial network (GAN) generated natural and realistic images, which improved the ability of you only look once version 3 (YOLOv3) to detect die defects. Then defects were measured based on the bounding boxes predicted by YOLOv3, which potentially provided the criteria for die quality sorting. The pseudo defective images generated by GAN from the real defective images were used as the training image set. The results obtained after training with the combination of the real and pseudo defective images were 7.33% higher in testing average precision (AP) and more accurate by one decimal place in testing coordinate error than after training with the real images alone. The GAN can enhance the diversity of defects, which improves the versatility of YOLOv3 somewhat. In summary, the method of combining GAN and YOLOv3 employed in this study creates a feature-free algorithm that does not require a massive collection of defective samples and does not require additional annotation of pseudo defects. The proposed method is feasible and advantageous for cases that deal with various kinds of die patterns.

Keywords: wafer die; defect detection; generative adversarial network (GAN); you only look once version 3 (YOLOv3)

1. Introduction

Wafer is the major material for making integrated circuits (ICs), and it plays an indispensable role in electronic products. The upstream of the semiconductor industry are IC design companies and silicon wafer manufacturing companies. IC design companies design circuit diagrams according to customer needs, while silicon wafer manufacturing companies use polysilicon as the raw material for silicon wafers. The primary task of IC manufacturing companies in the midstream is to transfer the circuit diagrams to wafers. The completed wafer is then sent to the downstream IC packaging and testing companies for packaging and testing the functions of ICs, concluding the whole manufacturing process.

With the continuous evolution of wafer manufacturing technology, wafer sizes have become larger and the patterns on the die have become more diverse. In order to inspect surface defects in the dies of a wafer, automated optical inspection (AOI), mainly using one or more optical imagery charge-coupled devices (CCDs), has gradually replaced traditional manual visual inspection (VI).

The current die AOI inspection methods of die inspection can be divided into design-rule checking [1,2], neural network [3–5], golden template method [6,7], and golden template self-generating method [8,9]. Design-rule checking is also known as the “knowledge database method”. Mital and Teoh [1] and Tobin et al. [2] found the geometric features and textural features of the components on the dies and stored them in the knowledge library, and then compared the features to determine whether they are defective or not. Neural network methods learn the mapping relationship between die features and defect classes throughout learning their synaptic weights. Su et al. [3] cut the die images into several windows of fixed size, and extracted the corresponding average gray value as features, and then established the back-propagation neural network (BPNN), the radial basis function neural network (RBFNN), and the learning vector quantization neural network (LVQNN) models. Chang et al. [4] used the kmeans to distinguish the P-electrode, the light-emitting area, and the background in the LED die. Then they extracted the geometric features and the textural features from the P-electrode and the light-emitting area, and distinguished the defects using the LVQNN. Timm and Barth [5] used radially encoded features to measure the discontinuity around the P-electrode of the LED die. Since there were relatively few defects, an anomaly detection method such as the one-class support vector machine (SVM) could obtain extremely high classification accuracy. However, the geometric features and appearance of various die images are quite different. It is difficult to find effective handcrafted features and establish criteria. Golden template method is also called the “image-to-image reference method”. This method compares the golden template with the image to be inspected, and the areas with significant differences are regarded as the ones where the defects may occur. Chou et al. [6] used the golden template method to highlight defects and measure the size, shape, location, and color of the defects, and then distinguished the types of defects based on design-rule checking. Zhang et al. [7] cut the boundary between the pads and the dies before establishing a golden template to highlight defects, and then determined the defect types based on features such as location, number of objects, and area of objects. This type of golden template method usually uses pixel-wise difference to identify the difference. Before performing the pixel-wise difference, the image to be inspected must be carefully aligned with the golden template. To alleviate alignment problem, golden template self-generating methods were introduced. Guan et al. [8] used the characteristics of repetitive patterns of the dies in the wafer to self-generate a defect-free template to highlight the defects. Liu et al. [9] used a discrete wavelet transform (DWT) to extract the standard image from three defective IC chip images, and then used the difference between the standard image and the defect image to highlight the defects. Liu et al. [9] also used the image gray-scale matching method to reduce the impact of different brightness on the detection results. This method avoided the alignment problem, but was limited by the need to zoom out to capture the die’s repetitive pattern when shooting the image. Since the image resolution was reduced, small defects became difficult to detect. However, the defect inspection algorithms used in AOI systems often need to be highly customized, highly accurate alignments, and rely heavily on experts to design hand-crafted features. As a result, existing algorithms described above cannot be applied to different product types [1–9]. A cruel truth is that die AOI inspection machines are often idle in practice when the production line is changed.

To overcome the above problem, deep learning methods have been introduced into die defect classification in recent years. Cheon et al. [10] proposed a four-layer convolutional neural network (CNN) architecture based on stacked convolution and max-pooling to classify five types of die defects. They also used the k nearest neighbor (KNN) in the 3D autoencoder output space to detect unknown defects that were very different from the existing types. Lin et al. [11] designed a six-layer CNN to classify LED chip defects, and used class activation mapping (CAM) to create a heat map corresponding to the analyzed image to locate the defective area. Chen et al. [12] constructed a CNN network based on the separable convolutional and bottleneck for six times to classify four types of die defects. From the existing literature, it finds that deep learning methods are shown not to require any feature extraction process, to be able to avoid shift, rotation, exposure, and so on, which is very powerful and has attracted global attention. However, most studies focused on how to use deep learning methods to

solve the problem of die defect classification, while relatively few focused on how to use those to solve the problem of die defect detection. The latter is the focus of the present study and makes a contribution to introduce an object detection method, you only look once version 3 (YOLOv3), to solve the die defect detection problem. The YOLOv3 model is able to predict the center coordinate, width, and height of each bounding box where the defect is located, and the confidence that each bounding box contains a defect. There is no need to rely on experts for feature engineering and have a certain degree of invariance to interference such as translation and rotation, which are attractive characteristics for companies that face constant changes die patterns. In addition, since the particle defects embedded on the dies are very tiny and some of the defects are dense, YOLOv3 uses DarkNet53 as the backbone and introduces multiscale detection, which is able to detect defects of different sizes on the extracted feature maps. In this way it can effectively detect tiny and dense defects, ensuring the quality of the die.

Moreover, there is another issue regarding defective samples collection and annotation in the factory environment. Operators do not have much time to collect various appearances of different kinds of defects. Since the collection of defect images is time-consuming, recent research on generating pseudo defective images with GAN has attracted attention. Chen et al. [12] uses affine transformation and naïve generative adversarial networks (GAN) to tackle the problem of having unbalanced quantities of defect-free and defective images. They expanded the number of defective images that enhanced the classifier's generalization ability. Tsai et al. [13] applied cycle-consistent adversarial network (CycleGAN) to generate the saw-mark defect in heterogeneous solar wafer images and to solve the unbalanced classification problem arising in manufacturing inspection. Their experiment showed that the CNN's classification accuracy rates of GAN-based data augmentation were better than those of doing over-sampling and assigning higher class-weights on minor classes. In addition to the research related to defect classification, the GAN-based method was also applied to the field of defect detection. Yang et al. [14] introduced an image generating process for welded joints, which was based on the affine transformation and CycleGAN. Then the YOLOv3 model was used for the welding head detection, with a better average precision (AP) of 91.02% than faster region-based convolutional neural network (Faster RCNN). Tian et al. [15] used CycleGAN to augment images of healthy apples and apples with anthrax, thereby increasing the number of images and enriching its content. After that, YOLOv3-dense was used to test the apple for anthracnose. Experiments showed that their model performs at an AP of 95.57%. This method could also be applied to the detection of apple surface diseases in orchards. However, some welded joints on metals or anthracoses on apples generated by CycleGAN are not expected output [14,15]. After augmenting the data using GAN, rich pseudo images are obtained. Although the appearance diversity of defects increases, no corresponding annotation files are provided [14,15], as operators have no time for time-consuming annotation work. Besides, GAN is currently unable to form specific structures, generative images that are not only blurry but also incorrectly colored. These undesirable generative images should be manually deleted. Cooperating GAN with an automatic annotation method is another contribution of this study, so that the data pairs are available for training the YOLOv3 die defect detection model. This research uses a series of pre- and post- digital image processing (DIP) techniques to reduce the generation loading of GAN and to develop an autoannotation procedure for pseudo defective images. The DIP techniques not only help to generate realistic pseudo defects but also save the time needed for annotating defective pseudo images.

This paper is composed of four parts. Following this introductory section, which has summarized the literature on die inspection and presents the contributions of this study, is the second section that describes the hardware architecture for capturing images, and the methodology. It also introduces GAN, the automatic annotation mechanism, and the modified application of YOLOv3. The third section presents the experimental results. A spot-checking process helps us to determine the YOLOv3 as the base model. The hyperparameters to be used in the GAN + YOLOv3 mechanism are derived, based on the design of the experiment (DOE). The defect detection results are reported, compared, and analyzed. The final part is the conclusion.

2. Research Method

The overall research process of this study is shown in Figure 1. First, we captured the images through the image-capturing system. We examined the die image structure and composition, and the appearance, characteristics, and specification of the particles. The next step was to separate the image set into training, validation, and testing sets. We manually marked the fine particles embedded on the surface of the die through an annotation tool to create the annotation file corresponding to each image. Defect-free dies do not need to be annotated and are not included in the training process. In order to increase the diversity of defects, the study created pseudo particle defects with the help of GAN's automatic generation ability. The study also automatically generated an annotation file corresponding to each pseudo defective image through the connected component labeling (CCL) [16]. The next step was to feed real and pseudo defective images to the YOLOv3 model for learning. Finally, we measured size of defects. Details of the research procedure are explained in the following sub-sections.

Figure 1. The research process.

2.1. Hardware Structure and Composition of the Die

In order to retrieve the surface images of the dies on the wafer, the study used the image-capturing system shown in Figure 2a. The CCD in this system was Hitachi KP-FD202GV, and the resolution was a 1620×1220 color image. The lens is an Olympus Lens with an optical magnification of $5\times$, a working distance of 19.6 mm, and a resolution of $3.36 \mu\text{m}$, coupled with a lighting source with a 12 V/100 W coaxial yellow ring halogen lamp. The front-illuminated light source emphasizes the surface characteristics of the inspection object. During the shooting process, the researchers used the

XY axis motion controller to capture the image of each die to be inspected with an S-shaped scanning path. By lowering the requirements for positioning precision, one image could contain multiple dies, as shown in Figure 2b. However, only the die pattern at the center of the image was intact, called region of interesting (ROI), and the eight neighboring dies had only partial patterns.

Figure 2. The image-capturing system and composition of the die. (a) The scanning path and field of view (b) the ROI die and (c) the appearance of die.

The appearance of the image of the die surface in this research is shown in Figure 2c. In compliance with a confidentiality agreement with the case company, images displayed in this paper are only part of the die, and the images have been flipped and discolored before presentation. The die was composed of the pad, the ion implantation zone, the bottom layer and the testing block. The pad was mainly used for electrical testing to ensure the function of the die. The bottom layer was a protective layer protected by a thin film, which could protect the components from chemical reactions, moisture, corrosion, pollution, etc. The testing block was used by the foundry customers to perform special tests. During the manufacturing process, the wafer might contaminate the surface of the die due to particle residues, which could result in defective products. These particles appeared at random positions, and they might be seen on the surface of the entire die. The shape of the particle was irregular, either large or small, and sometimes dense clusters occurred. The testing block on the die was a dark rectangular pattern with an appearance similar to that of particle defects, which increased the difficulty of defect detection.

2.2. Manually Annotating Defects

After building the die image set, we needed a corresponding annotation set before the model could be trained. Labelling was used as an annotating tool by the researchers to manually annotate the locations and names of the defects in the image one by one. These annotation messages were stored in the XML format, and the filename was the same as the filename of the annotated image, except for the filename extension. As shown in Figure 2b, only the central die in the image was the ROI. The traditional method might have been to design an algorithm to perform ROI image segmentation before proceeding to subsequent actions. However, since this study would use the object detection method in deep learning, the preprocess of extracting ROI could be omitted. As long as the researchers focused on framing the defects on the ROI die when annotating, later the algorithm would naturally ignore the defects on the eight neighboring dies when detecting defects.

2.3. Defect Data Augmentation by GAN and Their Autoannotation

As proposed by Goodfellow et al. [17], GAN has a wide range of applications, such as fashion, advertising, science, and games. Since the images for defect detection are usually captured in a stable environment, each image is roughly the same regardless of location or color. Therefore, the general traditional data augmentation method is not necessarily applicable. The case company does not have much time for engineers to collect a huge image set, let alone an additional time-consuming annotation work for a large number of image sets. To overcome this difficulty, this study took advantage of the powerful generative capabilities of GAN to create richer types of defects. The basic idea of GAN is shown in Appendix A.1 of Appendix A.

When we directly input a set of die images into the GAN model, however, we found that its objective function value fluctuated during the iteration and converged only with difficulty. Additionally, we also found that the GAN model could not generate high-resolution pseudo images effectively. It could only generate the approximate outline of the die, but the details could not be identified. Consequently, a strategy of simply generating the particle defects was adopted.

The detailed process is shown in Figure 3. We used the defect coordinate position, and the information of length and width that the annotator has previously noted in the real image, which helped to cut out the patch that indicates the area of each particle defect. Otsu binarization [18] was then used to eliminate the background in the patch as far as possible, retaining the original appearance of the particle defects, and the defects were attached to a white background image with the same size as the GAN input image. As shown in the bottom left of Figure 3, GAN is composed of two networks: a generator and a discriminator. During the first iteration, the generator generated poor pseudo images and the discriminator distinguished them from real images easily. During the second iteration, the quality of pseudo images generated by the generator was improved, which fooled the underlying discriminator. With the rise of the ability of the discriminator, real images and pseudo images can be recognized, which will also drive the improvement of the generator. The training process of adversarial learning between the two networks was continued and a generative model similar to the real image distribution was created. As the learning object became simpler, the objective function converged rapidly, and generated more realistic pseudo particle defects. Finally, we embedded the pseudo particle defects into the defect-free dies to create a generative pseudo image set.

Figure 3. The flowchart of GAN-based data argumentation and auto-annotation.

Although we used GAN for data augmentation to increase the diversity of defects, the annotation files of these pseudo defective images were not generated. In line with the previous literature, an additional manual method was adopted to annotate the pseudo defect images [14,15]. In order to save time when annotating the pseudo defective images, DIP techniques were used to automatically annotate the pseudo particles as shown in the bottom right of Figure 3. The CCL algorithm [16] scanned the image from left to right and top to bottom. If the gray values between adjacent pixels were found to be similar during scanning, they were labeled with a same index. Each pseudo particle defect would be regarded as a blob, and information of its minimum bounding box could also be registered. Then the XML annotation file of the pseudo defective image could be output, which reduced the time spent in annotating the pseudo defective images.

2.4. Defect Detection and Measurement Using YOLOv3

This research used YOLOv3 [19] as the basis for die defect detection and measurement. The basic idea of YOLOv3 is shown in A.2 of Appendix A. The YOLOv3 model is a one-stage method, end-to-end training process that can be realized using a single network. The inference can predict the center coordinate, width and height of each bounding box where the defect is located, and the confidence that each bounding box contains a defect.

After YOLOv3 output the predicted bounding boxes, the study would further measure the defects of the corresponding patches and sort the quality of a die, as shown in Figure 4. The process included Ostu binarization [18], the estimation of the bounding ellipse, and the calculation of the major and the minor axis. The process was able to potentially assist to conduct the sorting of the dies in accordance with the quality specifications of the customers. For example, there were three classes of die products: an excellent die had no particle defect after inspection; a qualified die had particle defects with the major axis length between 50 and 149 μm and the minor axis length less than 20 μm ; an unqualified die had particle defects that exceeded the quality specification.

Figure 4. Die defect detection and measurement.

3. Analysis and Results of the Experiments

This study collected 669 die images, 198 defect-free images, and 471 defective images with particles. Since training an object detection model needs only defect samples of the “object”, this research randomly selected 300 defective images as the training image set. The remaining defective images were used as testing image sets to evaluate the inference performance of the model. In addition, on the production line, the appearance of defects is multifaceted. It is not possible to produce distinctively different defect appearances if we only depend upon the jitter mechanism of YOLOv3 itself. In addition to the images generated by the jitter mechanism, this research also applied the GAN to generate images of pseudo die defects.

3.1. Spot-Checking Experiment

The spot-checking experiment gets a quick assessment of different models on a custom dataset. Researchers are able to know which type of models is suitable at picking out the structure of the

dataset. In order to demonstrate the performance of the object detection models for the detection of particle defects on the dies, this study compared the YOLOv3, the Faster RCNN [20], and the single shot multibox detector (SSD) [21]. After all the training processes were completed, the validation AP was used to evaluate the performance of the models on defective images. As shown in second column of Table 1, there were significant gaps of validation APs between YOLOv3 and other two models. In practice, the inference speed of a model was always concerned. The frames per second (FPS) was adopted here to evaluate the inference speed of the models as shown in the last column of Table 1. We found that the inference speed of SSD was fastest, followed by the YOLOv3 and lastly by faster RCNN. Even though the FPS of YOLOv3 was not the fastest, it was enough to be applied to the production line. The spot-checking experiment indicates that the YOLOv3 was the best model at learning the structure in the dataset so we could focus the attention to optimize it.

Table 1. Spot-checking comparison using different evaluation metrics.

Evaluation Metrics Methods	Validation AP	FPS
YOLOv3	81.59%	13.447
Faster RCNN	14.00%	2.783
SSD	57.56%	23.852

3.2. Hyperparameter Sensitivity Experiment

The hyperparameters of a model are related to the flexibility and potential of its learning, and directly influence the degree of the generalization when the model makes inferences. Since training a deep learning model often takes a long time, it is extremely inefficient to find the optimal hyperparameter combination manually for a deep learning model. Based on DOE, the research analyzed the validation AP with various hyperparameter combinations. It endeavored to identify the key hyperparameter combinations that affect the AP of die defect detection, which provides the basis for improving the AP of the model in a reasonable way. The DOE of this research includes four factors, and each factor has three levels.

- Input image size of GAN: The quality of the image generated by the GAN is not only influenced by the background complexity of the input image, but also by the size of the input image. If the size is too small, the GAN cannot generate detailed defects; if the size is too large, the GAN would weaken during the generation process. This research set three levels of the factor to be 28×28 , 64×64 , and 96×96 , with 28×28 as the default value, and 96×96 the upper limit of the defect size.
- Fold size of GAN image augmentation: This study proposed an image augmentation technology based on the GAN. Defect patches were generated from the adversarial learning process. Then the patches were pasted onto the defect-free die image to generate pseudo defect images. Compared with the original defective die images in Figure 5(a1,a2), the defects in Figure 5(b1,b2) were naturally embedded in the die image. This process generated various shapes, sizes and numbers of defects, and increased the quantity of training images and the diversity of defects, allowing YOLOv3 to learn a richer appearance of defects, and improving the performance of model training. In this study, the factors were set to three levels: 1, 1.5, and 2. The 1 meant that the GAN image augmentation mechanism was turned off; 1.5 and 2 meant that the original number of defective die images was multiplied by 1.5 and 2 respectively as the pseudo defective die images for later training.
- Upper limit of the input image size: There are only convolutional layers in YOLOv3, and its input image size is unrestricted. However, in order to strengthen the robustness of model inference, YOLOv3 adopted a multiscale training strategy. During the training process, the size of the input image was changed after a certain number of iterations. YOLOv3 could define the upper limit of the size range of the input image during the training process. In addition, because the minimum input feature map in Yoloblock employed downsampling 32 times, the upper limit of the image

size range must also be a multiple of 32. In the experimental design, the factor was set to three levels: 416×416 , 480×480 , and 544×544 . 416×416 was the default value of YOLOv3.

- Degree of jitter: In addition to the pseudo images generated by the GAN, the YOLOv3 also had its own data augmentation program, called the degree of jitter. It could flip, zoom, crop, and perform HSV contrast conversion of the input image to augment the images and suppress overfitting. This research set the factor to three levels: 0, 0.15, and 0.3. The 0.3 was the default value of YOLOv3, and 0 indicated that the jitter was turned off.

Figure 5. Comparison of defect images. (a1,a2) Original defects and (b1,b2) pseudo defects.

Next, this study removed 20% from the training image set to be used as the validation image set (not including any pseudo defective images). After conducting 3^4 DOE, the main effect plots of the validation AP for all the factor and level combinations were drawn as shown in Figure 6. Using the criterion “the larger the better” (LTB) for validation AP, the researchers selected the hyperparameter combination: the input image size and augmentation fold size of the GAN were 64×64 and 2, and the upper limit of the input image size and jitter degree of YOLOv3 were 416×416 and 0.3.

Figure 6. The main effect plot of 3^4 design of the experiment (DOE).

3.3. Results of Die Defect Detection and Measurement

After deciding the hyperparameters of the GAN + YOLOv3 model and training the model, defect detection and measurement of the remaining test images were performed. The pipeline of the testing process first inferred the predicted bounding boxes of the defects through YOLOv3. Then, the major and the minor axis of the defects were measured for the content inside the bounding box. After the inference was completed, different evaluation metrics were used to measure the generalization ability of the proposed algorithm.

The testing AP was used to measure the performance of the predicted bounding boxes: after the testing image set was inferred by the object detection method, the predicted bounding boxes were compared with the ground truth boxes, and the average of the maximum precision values calculated

when recall $\geq 0, 0.1, \dots, 1.0$. Coordinate prediction error was used to measure the accuracy of the coordinate prediction: after the testing image set was inferred by the object detection method, the closeness of the center coordinates, length, and width of the predicted bounding boxes were compared with those of the ground truth boxes, which could be calculated through the first two items in Equation (A2) of Appendix A.

Figure 7 demonstrates the patches of defect detection results. When inputting the die image of the product, as shown in Figure 7(a1–a4), the model precisely box-bounds the corresponding particles, as in Figure 7(b1–b4). The testing blocks in Figure 7(b1–b4) are not falsely box-bounded. YOLOv3 is able to discriminate between irregular-shaped particles and rectangular testing blocks. The model effectively detected particle defects on the surface of the die, and even very small defects could be detected successfully.

Figure 7. Defect detection results by GAN + YOLOv3. (a1–a4) Images to be inspected and (b1–b4) detection results.

In order to further demonstrate the performance improvement of the GAN-based image augmentation technology for the detection of particle defects on the dies, this research also constructed the YOLOv3 model, the GAN + YOLOv3 model (augmenting 1.5 times the training sample), the GAN + YOLOv3 model (augmenting 2 times the training sample), and the CycleGAN + YOLOv3 model (augmenting 1.5 times the training sample). After the training of the four models was completed, the study used the testing AP and the testing coordinate prediction error to evaluate the models on testing images.

Before calculating AP, the precision–recall (PR) curve of each model was drawn as shown in Figure 8. The pseudo defective die image generated by the GAN could work along well with the true defective die image to train the YOLOv3. In addition, when GAN helped to increase the training image by about 1.5 times, the PR curve tended to converge as shown in Figure 8, and the corresponding testing AP jumped from 81.39% to more than 88%, an increase of about 7% as shown in the second column of Table 2. As indicated by the results of the testing coordinate prediction error indicator, the coordinates and length and width of the predicted bounding boxes and the ground truth boxes, were very close to the ground truth labels. Even without the help of the GAN, the bounding box error predicted by the naïve YOLOv3 model was below three decimal places. After adding the GAN, the testing coordinate prediction error could be reduced to a level below four decimal places as shown in the last column of Table 2. This experiment shows that the pseudo defect images generated by the GAN play an important role in enriching the diversity of defects, which helps to improve the efficacy

and versatility of the model. Beside, we also compared the CycleGAN + YOLOv3 with proposed GAN + YOLOv3. The corresponding result is shown in the last row of Table 2. It clearly found that its testing AP and coordinate prediction error were not satisfied. The main reason is the appearance of the particle patches generated by CycleGAN was far from the real particle. Not only the area of defect was large, but also the edge of defect was not smooth.

Figure 8. Precision–recall (PR) curve for each method.

Table 2. Comparison of methods by different evaluation metrics.

Evaluation Metrics Methods	Testing AP	Testing Coordinate Prediction Error
YOLOv3	81.39%	1.6456×10^{-3}
GAN + YOLOv3 (1.5-fold increase)	88.12%	2.9662×10^{-4}
GAN + YOLOv3 (2-fold increase)	88.72%	1.5851×10^{-4}
CycleGAN + YOLOv3 (1.5-fold increase)	33.14%	6.4880×10^{-1}

4. Conclusions

Defect sample collection, defect annotation, and feature engineering have always been the most time-consuming tasks in defect detection. To address this issue, this research integrated the technology of generative pseudo defective samples (using GAN), automatic pseudo defect annotation (using DIP), and automatic feature extraction (using YOLOv3). The methods proposed in this study need not to rely on experts for feature engineering and did not need bulk defect samples. Massive defect annotations are not required, either. Users need only to prepare a few defect image sets, manually annotate them, and complete the model training before conducting the inferences. This means that the method has great potential for application in various die patterns, where appearances are changeable and complex. In addition, the experimental results show that after the addition of the GAN mechanism, both the overall detection precision of the predicted bounding box and the measurement accuracy of quality classification were improved. This indicates that the pseudo defect images generated by the GAN help enrich the diversity of the training data set, which to some extent improved the versatility of the model.

If semantic segmentation methods make a breakthrough in the inference speed in the future, it may be possible to combine the GAN and semantic segmentation methods to perform defect segmentation. The annotation process of the object segmentation model captures the outline of the defect in the

image, rather than simply annotating the rectangular bounding box, as happens in the object detection model. Therefore, the annotation does not contain the background and does not consider the angle. The annotation may include other rectangles near the defects. In this way, the process of removing the background and the process of extracting blobs from the predicted bounding box can be omitted, and the efficiency of model inference can be improved.

Author Contributions: Conceptualization, S.-H.C. and D.-B.P.; methodology, S.-H.C.; software, C.-H.K.; validation, S.-H.C., C.-H.K. and D.-B.P.; formal analysis, C.-H.K.; data curation, D.-B.P.; writing—original draft preparation, S.-H.C.; writing—review and editing, D.-B.P.; funding acquisition, S.-H.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ministry of Science and Technology, Taiwan, grant number MOST 108-2221-E-131-006-MY2 and The APC was funded by Ming Chi University of Technology.

Acknowledgments: We are grateful to three anonymous reviewers for comments. The authors acknowledge all participants, Yuan-Shuo Chang, Yu-Hsin Yen and Hsin-Chi Chang, for their domain knowledge and annotation support of the study.

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

Appendix A

We described the technical sections related to GAN and YOLOv3 in Appendix A.1 and Appendix A.2 respectively, so that the readers can focus on the important messages in the main texts.

Appendix A.1. Descriptions of the GAN

The network structure of GAN is shown in Figure A1 [17]. GAN is composed of two networks: a generator (G) and a discriminator (D). The generator is a four-layer neural network of regression. By obtaining the distribution of the real image and inputting the noise source ($\mathbf{z}$) to the generator, it produces a pseudo image similar to the real one. The discriminator is a three-layer neural network of the binary classifier, which is responsible for evaluating the authenticity of the pseudo image. With the rise of the ability of the discriminator, real images, and simulated images can be recognized, which will also drive the improvement of the generator. Subsequently the generator generates pseudo images closer to real ones for the discriminator to distinguish. Finally, the generator can generate pseudo images very similar to real ones. Between the two networks described above, the training process of adversarial learning is continued, interactive learning is realized, and a generative model similar to the real image distribution is created. The loss function of the model is shown in Equation (A1):

$$\min_G \max_D V(D, G) = E_{\mathbf{x} \sim P_{data}(\mathbf{x})} [\log D(\mathbf{x})] + E_{\mathbf{z} \sim P_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))] \quad (\text{A1})$$

where $\mathbf{x}$ is an image from the real data distribution P_{data} ; $\mathbf{z}$ is a noise vector sampled from a uniform distribution or a normal distribution P_z ; and E represents the expectation of real data and that of noise.

Figure A1. Architecture of the GAN network.

Appendix A.2. Descriptions of the YOLOv3

The network structure of YOLOv3 is shown in Figure A2. With DarkNet53 as its backbone, the model used a series of 1×1 and 3×3 convolution layers for feature extraction without a pooling layer and a fully connected layer. YOLOv3 introduced a residual block, which adds the corresponding dimensions of the input and the output feature maps to control the magnitude of the gradient propagate and alleviate the vanishing gradient problem faced by the deep network. In addition, the feature pyramid network (FPN) structure was used for multiscale detection effects. After the input image passed through DarkNet53, the feature map generated by Yoloblock was taken for two purposes. The first was to generate feature map 1 with a size of 13×13 after passing through the 3×3 and the 1×1 convolution layer. The second use was to add an upsampling layer after passing through the 1×1 convolutional layer, and to splice it with the output result of the intermediate layer of the DarkNet53 network, which generated feature map 2 with a size of 26×26 . After the same loop, feature map 3 with a size of 52×52 was generated. 13×13 , 26×26 , and 52×52 are the number of grid cells of the output feature maps for each scale. The depth of these output feature maps, was set as $B \times 5$, where B represented the number of predicting bounding boxes for each scale, set here as 3. The number 5 represented the 5 levels of x, y, w, h , and confidence that must be predicted for each bounding box, where x and y represent the shift levels between the predicted bounding box center and the upper left corner of the grid cell; w and h represent the ratio of the width and height of the predicted bounding box to the width and height of the entire image; confidence represents the confidence value of the defect. The depth of the feature map output for traditional YOLOv3 needs to include the probability of the predicted bounding box of each grid cell for C categories, so its depth should be $B \times (5 + C)$. However, this research only sought to classify the single problem of particle defects, so the prediction of C could be omitted.

Figure A2. Architecture of the YOLOv3 network.

During training, YOLOv3 used the modified loss function and the back-propagation algorithm to learn the weights, as shown in Equation (A2). The loss function of YOLOv3 was originally composed of three parts, namely coordinate prediction error, intersection over union (IoU) error, and classification error [19]. However, since this research focused on the single classification problem, the classification error could be omitted.

$$\text{Loss} = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbf{I}_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbf{I}_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \\ + \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbf{I}_{ij}^{obj} (C_i - \hat{C}_i)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbf{I}_{ij}^{noobj} (C_i - \hat{C}_i)^2 \quad (\text{A2})$$

The first two terms in Equation (A2) represent the coordinate prediction error, and λ_{coord} was the weight hyperparameter given in advance. Since the number of grid cells that did not contain objects far exceeded the number of grid cell that contained objects, the loss of confidence without objects would be great. In order to reduce the impact of this problem on the network, it was generally assumed to be 5. $\mathbf{I}_{ij}^{obj}$ described the predicted bounding box j of the grid cell i had an indicator function containing objects. The $\hat{x}_i$, $\hat{y}_i$, $\hat{w}_i$, and $\hat{h}_i$ represent the central coordinates and the length and width of the i th predicted bounding box and the x_i , y_i , w_i , and h_i represented those of the i th ground truth box.

In addition, the last two terms in Equation (A2) represented the IoU error, where λ_{noobj} was the weight hyperparameter given in advance, which generally defaulted to 0.5. $\mathbf{I}_{ij}^{noobj}$ represented that the predicted bounding box j of the grid cell i did not contain the indicator function of the object. The $\hat{C}_i$ represented the i th predicted value of confidence; the C_i referred to whether the i th ground truth box contained object.

References

1. Mital, D.; Teoh, E. Computer based wafer inspection system. In Proceedings of the Proceedings IECON '91: 1991 International Conference on Industrial Electronics, Control and Instrumentation, Kobe, Japan, 28 October–1 November 1991; Institute of Electrical and Electronics Engineers (IEEE): New York, NY, USA, 2002; pp. 2497–2503.
2. Tobin, K.W.J.; Karnowski, T.P.; Lakhani, F. Integrated applications of inspection data in the semiconductor manufacturing environment. In *Metrology-based Control for Micro-Manufacturing*; Tobin, K.W.J., Lakhani, F., Eds.; SPIE: Bellingham, WA, USA, 2001; Volume 4275, pp. 31–40.
3. Su, C.-T.; Yang, T.; Ke, C.-M. A neural-network approach for semiconductor wafer post-sawing inspection. *IEEE Trans. Semicond. Manuf.* **2002**, *15*, 260–266. [CrossRef]
4. Chang, C.-Y.; Chang, C.-H.; Li, C.-H.; Jeng, M. Learning Vector Quantization Neural Networks for LED Wafer Defect Inspection. In Proceedings of the Second International Conference on Innovative Computing, Informatio and Control (ICICIC 2007), Kumamoto, Japan, 5–7 September 2007; Institute of Electrical and Electronics Engineers (IEEE): New York, NY, USA; p. 229.
5. Timm, F.; Barth, E. Novelty detection for the inspection of light-emitting diodes. *Expert Syst. Appl.* **2012**, *39*, 3413–3422. [CrossRef]
6. Chou, P.B.; Rao, A.R.; Sturzenbecker, M.C.; Wu, F.Y.; Brecher, V.H. Automatic defect classification for semiconductor manufacturing. *Mach. Vis. Appl.* **1997**, *9*, 201–214. [CrossRef]
7. Zhang, J.-M.; Lin, R.-M.; Wang, M.-J.J. The development of an automatic post-sawing inspection system using computer vision techniques. *Comput. Ind.* **1999**, *40*, 51–60. [CrossRef]
8. Guan, S.-U.; Xie, P.; Li, H. A golden-block-based self-refining scheme for repetitive patterned wafer inspections. *Mach. Vis. Appl.* **2003**, *13*, 314–321. [CrossRef]
9. Liu, H.; Zhou, W.; Kuang, Q.; Cao, L.; Gao, B. Defect detection of IC wafer based on two-dimension wavelet transform. *Microelectron. J.* **2010**, *41*, 171–177. [CrossRef]
10. Cheon, S.; Lee, H.; Kim, C.O.; Lee, S.H. Convolutional Neural Network for Wafer Surface Defect Classification and the Detection of Unknown Defect Class. *IEEE Trans. Semicond. Manuf.* **2019**, *32*, 163–170. [CrossRef]

11. Lin, H.; Li, B.; Wang, X.; Shu, Y.; Niu, S. Automated defect inspection of LED chip using deep convolutional neural network. *J. Intell. Manuf.* **2019**, *30*, 2525–2534. [\[CrossRef\]](#)
12. Chen, X.; Chen, J.; Han, X.-G.; Zhao, C.; Zhang, D.; Zhu, K.; Su, Y. A Light-Weighted CNN Model for Wafer Structural Defect Detection. *IEEE Access* **2020**, *8*, 24006–24018. [\[CrossRef\]](#)
13. Tsai, D.-M.; Fan, M.; Huang, Y.-Q.; Chiu, W.-Y. Saw-Mark Defect Detection in Heterogeneous Solar Wafer Images using GAN-based Training Samples Generation and CNN Classification. In Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, Prague, Czech Republic, 25–27 February 2019; pp. 234–240.
14. Yang, L.; Liu, Y.; Peng, J. An Automatic Detection and Identification Method of Welded Joints Based on Deep Neural Network. *IEEE Access* **2019**, *7*, 164952–164961. [\[CrossRef\]](#)
15. 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. Sensors* **2019**, *2019*, 1–13. [\[CrossRef\]](#)
16. Dillencourt, M.B.; Samet, H.; Tamminen, M. A general approach to connected-component labeling for arbitrary image representations. *J. ACM* **1992**, *39*, 253–280. [\[CrossRef\]](#)
17. Goodfellow, I.; Pouget, A.J.; Mirza, M.; Xu, B.; Warde, -F.D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. In *Advances in Neural Information Processing Systems*; Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., Weinberger, K.Q., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2014; Volume 27, pp. 2672–2680.
18. Otsu, N. A Threshold Selection Method from Gray-Level Histograms. *IEEE Trans. Syst. Man, Cybern.* **1979**, *9*, 62–66. [\[CrossRef\]](#)
19. Redmon, J.; Farhadi, A. Yolov3: An Incremental Improvement. 2018. Available online: <https://pjreddie.com/media/files/papers/YOLOv3.pdf> (accessed on 27 November 2020).
20. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards realtime object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *39*, 1137–1149. [\[CrossRef\]](#) [\[PubMed\]](#)
21. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. SSD: Single shot multibox detector. Computer Vision—ECCV 2016, Proceedings of The 14th European Conference on Computer Vision, Amsterdam, The Netherlands, 8–16 October 2016; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer: Cham, Germany, 2016; pp. 21–37.

Publisher’s Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).