Mathematical Biosciences and Engineering

MBE, 18(4): 4411–4428.
DOI: 10.3934/mbe.2021223
Received: 07 April 2021
Accepted: 13 May 2021
Published: 21 May 2021

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

Research article

Applying deep learning to defect detection in printed circuit boards via a newest model of you-only-look-once

Venkat Anil Adibhatla[1], Huan-Chuang Chih[2], Chi-Chang Hsu[2], Joseph Cheng[2], Maysam F. Abbod[3] and Jiann-Shing Shieh[1,*]

[1] Dept. Mechanical Engineering, Yuan Ze University, Chung-Li, Taiwan
[2] Dept. Advanced manufacturing system, Boardtek Electronics Corporation, Taiwan
[3] Dept. Electronic and Electrical Engineering, Brunel University London, Uxbridge, UK

* **Correspondence**: Email: jsshieh@saturn.yzu.edu.tw.

Abstract: In this paper, a new model known as YOLO-v5 is initiated to detect defects in PCB. In the past many models and different approaches have been implemented in the quality inspection for detection of defect in PCBs. This algorithm is specifically selected due to its efficiency, accuracy and speed. It is well known that the traditional YOLO models (YOLO, YOLO-v2, YOLO-v3, YOLO-v4 and Tiny-YOLO-v2) are the state-of-the-art in artificial intelligence industry. In electronics industry, the PCB is the core and the most basic component of any electronic product. PCB is almost used in each and every electronic product that we use in our daily life not only for commercial purposes, but also used in sensitive applications such defense and space exploration. These PCB should be inspected and quality checked to detect any kind of defects during the manufacturing process. Most of the electronic industries are focused on the quality of their product, a small error during manufacture or quality inspection of the electronic products such as PCB leads to a catastrophic end. Therefore, there is a huge revolution going on in the manufacturing industry where the object detection method like YOLO-v5 is a game changer for many industries such as electronic industries.

Keywords: convolution neural network; YOLO-v5; deep learning; printed circuit board (PCB)

1. Introduction

A Printed Circuit Board (PCB) is the core and primary section of any electronic product. As we observe in our day to day life we human beings are being reliable on electronic product, from

communication to transport, each and every product is being induced with PCB. These PCB are the core and base of any electronic product, which has to be designed and manufactured with a high precision to cope with demands and requirements keeping the quality up to the mark is really a challenging job. PCB are basically a base for the chips, transistors, capacitors and other electronic components which is made of fiberglass, composite epoxy with laminated materials [1–4]. During the manufacturing process many PCBs are produced with defects due to mishandling or technical faults during production. These faulty or defective PCBs should be detected and segregated during the quality inspection process. In the past there were many traditional and standard methods to detect defects [5,6] but they are not so efficient and fast.

In the recent couple of years, there is a lot of research done on detecting defects in PCBs but they were not so effective and unfortunately not able to detect tiny defects [7]. Although nowadays the PCB manufacturing industries use different image processing methods such as image subtraction, they are still not able to keep up with the quality inspection process. With the hike in popularity of consumer electronics products, accurate PCB manufacturing is important [8]. Few examples of PCB defect detection methods can be found in the literature [8–10]. Particularly, template-matching method [11] is used to detect defects in PCBs. Image subtraction method is another method using OpenCV [12]. However, these detection algorithms are specific to particular types of defects in PCBs. Remarkable progress has been made in the use of convolutional neural networks (CNNs) in several applications [13,14], such as image recognition [15,16] and object detection. In particular, object detection is achieved by implementing object recognition methods [17] and region-based CNNs [18].

These methods are used with the involvement of AOI and AVI machines. Apart from using such an expensive machine, PCB industries are bound to train and involve a huge amount of manpower for quality inspection after the traditional inspection process [19–21]. These industries need to rely on trained skilled manpower where there is an inconsistency in accuracy, and it consumes more time which delays the production. To overcome such kinds of difficulties and issues, YOLO-v5 has been introduced in this paper to overcome the difficulties that were experienced in previous research. However, training skilled engineers requires real skilled training. Even after years of training, human being makes errors. Eventually, the application of machine learning can reduce the error to some extent. Machines programmed with deep learning algorithms can be used to verify hard to detect defects.

Comparative to skilled quality engineer, machine learning methods are more accurate and much faster. Thus, this research proves that the involvement of machine learning can detect defects in PCB and increase productivity with higher accuracy. Researchers have applied various You-Only-Look-Once (YOLO) [22–26] approaches to art datasets and achieved excellent accuracy. YOLO-v5 outperforms other object detection algorithms due to its unique features like Mosaic data enhancement and adaptive anchor frame calculation. Apart from its well-designed structure, it outperforms in speed and it is smaller in size. On Ubuntu operating system, running a Titan V, we saw inference times up to 0.007 seconds per image, meaning 140 frames per second (FPS). By contrast, YOLO-v4 achieved 50 FPS after having been converted to the same PyTorch library. YOLO-v5 has several differences and improvements. In this research, we have used the PyTorch library to deploy the YOLO-v5 model which is a lot user-friendly for developers.

YOLO-v5 model can be also deployed in mobile devices as it can be compiled to CoreML and ONNX. 140 FPS has been achieving by YOLO-v5. When batch size is set to 1, 30 FPS as output is achieved by YOLO-v5 and 10 FPS is achieved by YOLO-v4. In our tests, we achieved roughly 0.895 mean average precision (mAP) after training for initial 100 epochs. Admittedly, we saw a comparable performance from EfficientDet and YOLO-v4, but it is rare to see such across-the-board performance improvements without any loss inaccuracy. Finally, YOLO-v5 is small in size. The weight file for

YOLO-v5 is about 27 megabytes and the weight file for YOLO-v4 is 244 megabytes. YOLO-v5 is nearly 90% smaller than YOLO-v4. This means YOLO-v5 can be deployed to embedded devices much more easily.

2. Materials and methods

2.1. PCB data set

Initially, data is collected from the AVI machine, the AVI machine provides an RGB format of the panel image of 4 different cameras. Images from 4 different cameras are extracted separately and then further process by cropping and testing which is done separately on the 4 different cameras. So initially, the R, G, and B image is combined and cropped to the exact location of the defect that is provided by AOI machine in a text file. The images is cropped to a size of 400 × 400 of the location and saved. This method is used to collect around 23,000 defective PCB images. After collecting the images, they are labelled using a tool created for the quality inspection engineers. This tool is designed to extracts the images from AVI and AOI machine of defective PCB with coordinates and label the defects. The quality inspection engineer labels the defective region with box shape around the defective region and label it as DEFECT eventually the location of two corners of the box which is x1y1 and x2y2 are saved in txt format which is used for the training process. After collecting and labeling the images, three different models like YOLO-v5 small, YOLO-v5 medium, and YOLO-v5 large are trained and their results are compared.

A total of 23,000 images are used in this experiment which is divided into 20,700 images for training and 2300 images for testing purposes. After training for all the three models using 10 cross-validation method, 30 models are generated and tested. The quality control testing procedure is fully automated with a user interface that provides the time and number of negative and OK images. This visualization user interface is effective and helps in monitoring the testing process without human interaction. The interface is developed using Python and Tkinter library. Figure 1 shows a snapshot of the developed interface used by the quality inspection operators.

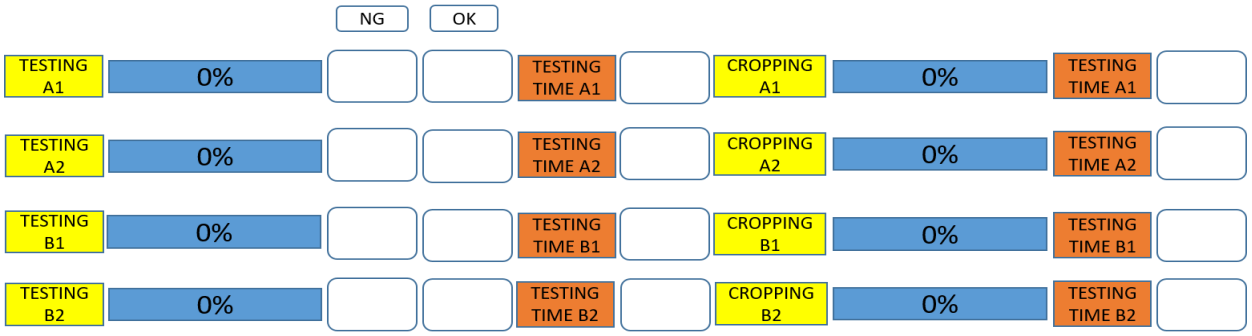


Figure 1. Snapshot of a user interface for testing monitoring and visualization.

2.2. Architecture of YOLO-v5

The structure of YOLO-v5 and YOLO-v4 very similar, but there are some differences. Figure 2

shows YOLO-v5 network structure diagram, it is still divided into 4 parts, namely: Input, Backbone, Neck, Prediction.

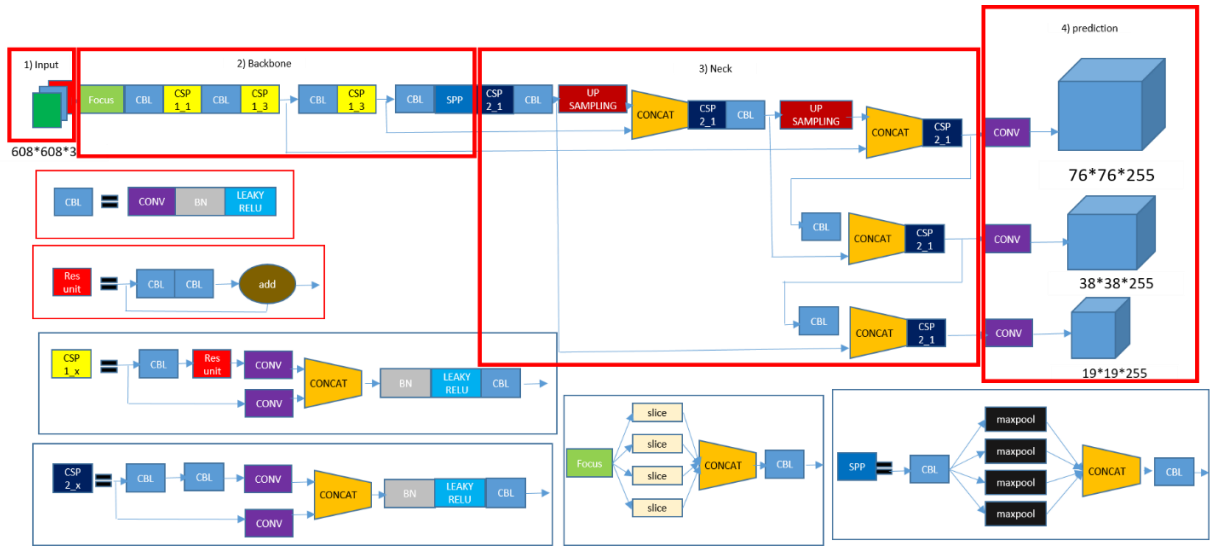


Figure 2. Structure of YOLO-v5.

2.1.1. Input

a) Mosaic data enhancement

The input end of YOLO-v5 uses the same Mosaic data enhancement method. During the usual project training, Arithmetic progression with small target Generally lower than the medium and large goals. Our data set also contains a large number of small targets, but the more troublesome is the distribution of small targets Uneven. There are several advantages, Rich data set, Random use of pictures, Random scaling, and random distribution for splicing, which greatly enriches the detection data set, especially random scaling adds a lot of small targets, making the network more robust. Reducing GPU, some people may say that random scaling, ordinary data enhancement can also be done. But the author considers that many people may only have a GPU, so Mosaic enhancement training can directly calculate the data of 4 pictures such that the Mini-batch size does not need to be very large. Therefore, a GPU can achieve better results.

b) Adaptive anchor frame calculation

In the YOLO algorithm, for different data sets, there will be an Anchor frame with initial length and width. During the network training stage, the network outputs are the prediction frame based on the initial anchor frame, then the sum of the Ground Truth Compare is calculated using the difference between the two, and then Iterative network parameters are updated in reverse. In YOLO-v3 and YOLO-v4, when training with different data sets, the calculation of the initial anchor box value is executed through a separate program. But in YOLO-v5, this function is embedded in the code. Hence, the best anchor box value is calculated during each training stage and updated adaptively.

c) Adaptive image scaling

In commonly used target detection algorithms, different pictures have different lengths and widths, so the common way is to uniformly scale the original pictures to a standard size, and then feed them to the detection network. During inspection, many defect images might have different aspect ratios, and after zooming and filling, the black image borders can be different. If more fillings is needed, there will be information redundancy, which will affect the speed of reasoning. Therefore, in the YOLO-v5 code, the letterbox function is modified to the original image and the least black border adaptively which is different with our previous study [27]. The black edges at both ends of the image height are reduced which consequently reduced the calculations during inference which improves the target detection speed. Through this simple improvement, the inference speed has been increased by 37%, which can be said to be very effective.

2.2.2. Backbone

a) Focus structure

Taking the structure of YOLO-v5 as an example, an original 608 × 608 × 3 image is fed into the Focus structure as shown in Figure 3, then the slicing operation is used to change the images to 304 × 304 × 12 feature map, followed by 32 convolution operation kernels which produces a final feature map of 304 × 304 × 32. The Focus structure of YOLO-v5 is shown in Figure 3.

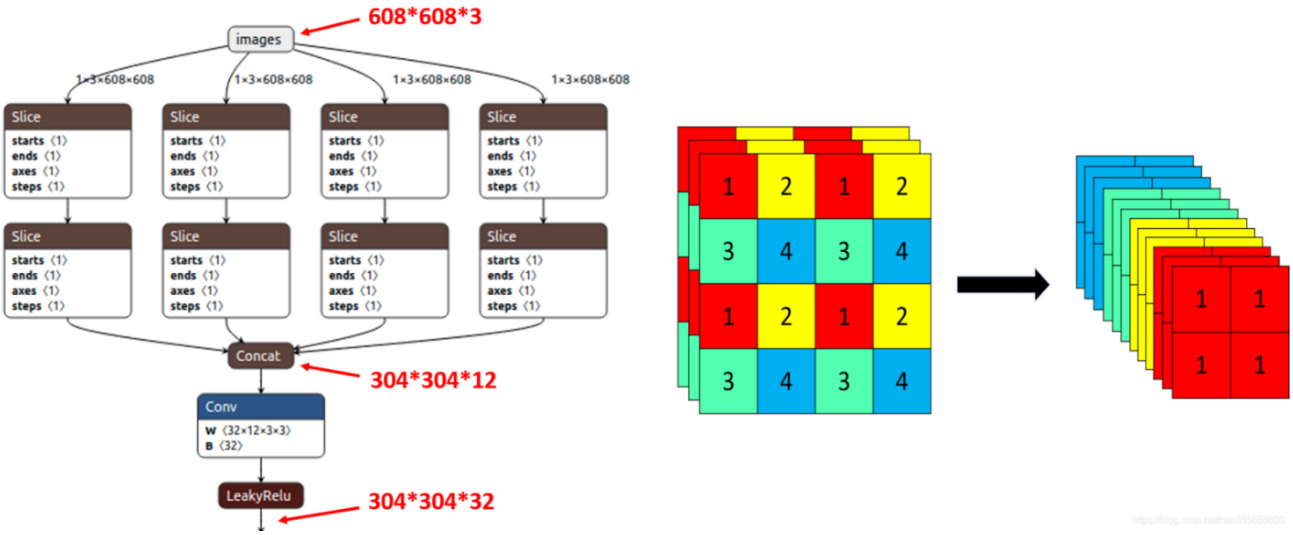


Figure 3. Focus structure of YOLO-v5.

b) CSP structure

The full name of CSP Net is Cross Stage Partial Network, which mainly solves the problem of a large amount of calculation in reasoning from the perspective of network structure design. The author of CSP Net believes that the problem of excessive inference calculations is due to network optimization gradient information repetition. Therefore, the CSP module is used to first divide the feature map of

the base layer into two parts, and then merge them through a cross-stage hierarchy, which can reduce the amount of calculation and ensure accuracy. There are two CSP structures designed in the YOLO-v5 network, for example, the CSP1_X structure applied to the Backbone network and an additional CSP2_X Structure is applied to Neck.

2.2.3. Neck

YOLO-v5's current Neck and YOLO-v4 use FPN + PAN structure, but when YOLO-v5 first came out, only the FPN structure was used, and the PAN structure was added later, and other parts of the network were also adjusted. The Neck structure of YOLO-v5 as shown in Figure 4, the CSP2 structure designed by CSPnet is adopted to strengthen the ability of the network feature integration.

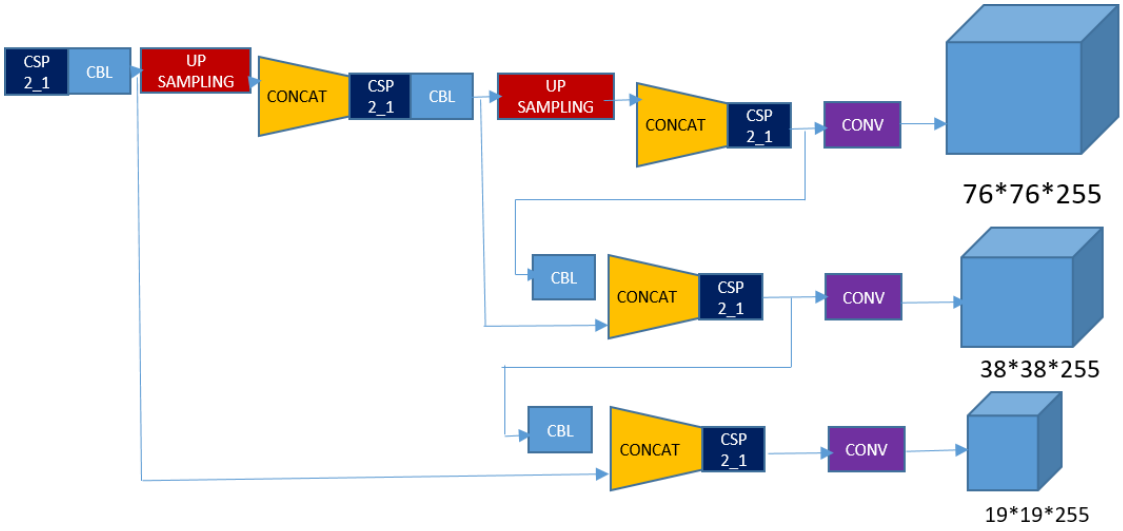


Figure 4. Neck structure of YOLO-v5.

2.2.4. Output

a) Bounding box loss function

YOLO-v5 uses Generalized Intersection over Union (GIOU) Loss as the loss function of the bounding box. In GIOU the measurement method of the intersecting scale is added which relieves the embarrassment of pure Intersection over Union (IOU). Based on the IOU, it solves the problem when the bounding boxes do not coincide.

b) Non-maximum suppression

In the post-processing of target detection, screening of many target frames usually requires a non-maximum suppression operation. In this research, DIOU_non-maximum suppression method is adopted which is different from our previous study [27]. Under the same parameters, the IOU in non-maximum suppression was changed to DIOU_non-maximum suppression. For some block overlapping targets, there will be indeed some improvements. When the parameters are consistent with

the ordinary IOU_non-maximum suppression, we modify it to DIOU_non-maximum suppression, and the two targets can be detected. Although the effect is similar in most states without an increase in the calculation cost, there is a slight improvement which is also good.

3. Results

In comparison with other object detection algorithms, the implementation of YOLO-v5 into embedded devices is very easy. Nvidia TITAN V GPU is used for this experiment, which reduces training time to 10% (i.e., 34 to 4 h). YOLO-v5 only requires the installation of a torch and some lightweight python libraries. With NVidia TITAN V GPU using Linux operating system and with the help of PyTorch, the experimental costs have been reduced because we just need to install the Torch with lightweight libraries. YOLO-v5 can infer individual images, batch images, video feeds, or webcam ports. The file folder layout is intuitive and easy to navigate while developing. You can easily translate YOLO-v5 from PyTorch weights to ONXX weights to CoreML to IOS. Three types of YOLO-v5 models are used in this experiment and have been compared with each other YOLO-v5 small, medium and large models. They have trained and tested the network setting, and parameters have been adjusted and tuned gradually using trial and error method. YOLO-v5 includes four different models ranging from the smallest YOLO-v5 with 7.5 million parameters (plain 7 MB, COCO pre-trained 14 MB) and 140 layers to the largest YOLO-v5 x with 89 million parameters and 284 layers (plain 85 MB, COCO pre-trained 170 MB). The approach that is considered in this paper is based on pre-trained YOLO-v5x model.

Table 1. Accuracy, misclassification rate, true positive rate, false positive rate, true negative rate, precision, prevalence, mean and standard deviation of accuracy for YOLO-v5 small model.

YOLO-v5 small model							
Category	Accuracy	Misclassification Rate	True Positive Rate	False Positive Rate	True Negative Rate	Precision	Prevalence
Crossvalidation 1	97.60%	0.023	0.971	0.02	0.97	0.97	0.45
Crossvalidation 2	97.56%	0.024	0.972	0.02	0.97	0.97	0.45
Crossvalidation 3	97.43%	0.025	0.972	0.02	0.97	0.97	0.45
Crossvalidation 4	97.56%	0.024	0.973	0.02	0.97	0.97	0.45
Crossvalidation 5	97.60%	0.023	0.971	0.02	0.97	0.97	0.45
Crossvalidation 6	97.65%	0.023	0.973	0.02	0.97	0.97	0.45
Crossvalidation 7	97.47%	0.025	0.970	0.02	0.97	0.97	0.45
Crossvalidation 8	97.56%	0.024	0.971	0.02	0.97	0.97	0.45
Crossvalidation 9	97.60%	0.023	0.971	0.02	0.97	0.97	0.45
Crossvalidation 10	97.39%	0.026	0.969	0.02	0.97	0.97	0.45
Mean ± SD	97.52 ± 0.07						

*Note. Accuracy: Overall, how often is the detection correct. (TP + TN) / total; Misclassification Rate: Overall, how often is it wrong. (FP + FN) / total; True Positive Rate: When it's actually yes, also known as "Sensitivity" or "Recall"; False Positive Rate: When it's no. FP/actual no; True Negative Rate: When it's actually no, also known as

"Specificity". TN/actual no; Precision: When it predicts yes. TP/predicted yes; Prevalence: It defines how often does the yes condition occurs in our sample. Actual yes/total.

Table 2. Accuracy, misclassification rate, true positive rate, false positive rate, true negative rate, precision, prevalence, mean and standard deviation of accuracy for YOLO-v5 medium model.

YOLO-v5 medium model							
Category	Accuracy	Misclassification Rate	True Positive Rate	False Positive Rate	True Negative Rate	Precision	Prevalence
Crossvalidation 1	99.17%	0.008	0.99	0.007	0.99	0.99	0.45
Crossvalidation 2	99.08%	0.009	0.99	0.008	0.99	0.99	0.45
Crossvalidation 3	99.17%	0.008	0.99	0.007	0.99	0.99	0.45
Crossvalidation 4	99.17%	0.008	0.99	0.009	0.99	0.99	0.45
Crossvalidation 5	99.17%	0.008	0.99	0.007	0.99	0.99	0.45
Crossvalidation 6	99.21%	0.007	0.99	0.007	0.99	0.99	0.45
Crossvalidation 7	99.20%	0.007	0.99	0.007	0.99	0.99	0.45
Crossvalidation 8	99.17%	0.008	0.99	0.007	0.99	0.99	0.45
Crossvalidation 9	99.13%	0.008	0.99	0.008	0.99	0.99	0.45
Crossvalidation 10	99.17%	0.008	0.99	0.009	0.99	0.99	0.45
Mean ± SD	99.16 ± 0.03						

*Note. Similar to Table 1.

Table 3. Accuracy, misclassification rate, true positive rate, false positive rate, true negative rate, precision, prevalence, mean and standard deviation of accuracy for YOLO-v5 large model.

YOLO-v5 large model							
Category	Accuracy	Misclassification Rate	True Positive Rate	False Positive Rate	True Negative Rate	Precision	Prevalence
Crossvalidation 1	99.86%	0.001	0.99	0.0008	0.99	0.99	0.45
Crossvalidation 2	99.82%	0.001	0.99	0.0016	0.99	0.99	0.45
Crossvalidation 3	99.82%	0.001	0.99	0.0008	0.99	0.99	0.45
Crossvalidation 4	99.78%	0.002	0.99	0.0002	0.99	0.99	0.45
Crossvalidation 5	98.86%	0.001	0.99	0.0008	0.99	0.99	0.45
Crossvalidation 6	99.86%	0.001	0.99	0.0008	0.99	0.99	0.45
Crossvalidation 7	99.95%	0.0004	0.99	0.0000	1.00	1.00	0.45
Crossvalidation 8	99.78%	0.002	0.99	0.0031	0.99	0.99	0.45
Crossvalidation 9	99.82%	0.001	0.99	0.0000	1.00	1.00	0.45
Crossvalidation 10	99.86%	0.001	0.99	0.0023	0.99	0.99	0.45
Mean ± SD	99.74 ± 0.29						

* Note. Similar to Table 1.

Table 4. Confusion matrix of five different cross-validations for YOLO-v5 small, medium and large.

			Real					
			S		M		L	
	model		NG	OK	NG	OK	NG	OK
Predicted	Cross validation 1	NG	1025	25	1040	10	1049	1
		OK	30	1220	9	1241	2	1248
	Cross validation 2	NG	1023	27	1039	11	1048	2
		OK	29	1221	10	1240	2	1248
	Cross validation 3	NG	1020	30	1041	9	1049	1
		OK	29	1221	10	1240	3	1247
	Cross validation 4	NG	1022	28	1038	12	1047	3
		OK	28	1222	7	1243	2	1248
	Cross validation 5	NG	1025	25	1040	10	1049	1
		OK	30	1220	9	1241	2	1248
	Cross validation 6	NG	1024	26	1040	10	10491	1
		OK	28	1222	8	1242	2	1248
	Cross validation 7	NG	1023	27	1041	9	1050	0
		OK	31	1219	8	1242	1	1249
	Cross validation 8	NG	1024	26	1040	10	1046	4
		OK	30	1220	9	1241	1	1249
	Cross validation 9	NG	1025	25	1040	10	1050	0
		OK	30	1220	10	1240	4	1246
	Cross validation 10	NG	1022	28	1038	12	1047	3
		OK	32	1218	7	1243	0	1250

YOLO-v5x model uses a two-stage detector that consists of a Cross Stage Partial Network (CSPNet) backbone, and a model head using a Path Aggregation Network (PANet) for instance segmentation. Each Bottleneck CSP unit consists of two convolutional layers with 1 × 1 and 3 × 3 filters. The backbone incorporates a Spatial Pyramid Pooling network (SSP), which allows for dynamic input image size and is robust against object deformations. In this experiment, we have used 23,000 images which are labeled by skilled quality control engineer different types of defects has been labeled as a single class as DEFECT. The epoch size is based on the training dataset. After deciding the parameters for the model, an initial ideal starts for training beginning. A cross-validation method has been induced to validate the results and 10 cross-validation method has been used. Then data is divided

into 10 parts, 9 of them are used for training and the remaining one part is used for testing.

The 10 cross-validation method is used for 3 different models which generates 30 cross-validation models to justify the result. The 10-fold cross-validation method [28] has been implemented to evaluate the execution of the trained models. Initially, the data is randomly divided into 10 equal parts, 9 of these parts are used for the training model and the remaining part is used for testing. After every training of the YOLO-v5 model, the data used to evaluate the model is not seen or interacted with the model during the training season. This method is repeated 10 times by jumbling the training and validation datasets. After completing the training process for the Tiny-YOLO-v5 model, every model is tested for different datasets.

As demonstrated in Tables 1–3, the results are gradually improving as the structure changes. This performance also proves that YOLO-v5 model is more efficient than other YOLO models. After every epoch or iteration, there is an increase in accuracy of the training process, which eventually tends the model to move towards better performance, and the final model is saved after the accuracy reaches a stable state. Results of 10 cross-validations YOLO-v5 small model are shown in Table 1, for YOLO-v5 medium model are shown in Table 2 and YOLO-v5 large model are shown in Table 3.

Table 4 displays the result of testing in the form of a confusion matrix. In detail, initially, YOLO-v5 small model detection accuracy is approximately 97.52%, the YOLO-v5 medium model's accuracy is 99.16%, YOLO-v5 large model is approximately 99.74% as reported in Tables 1–3 respectively. In total, 30 cross-validations have been trained on 3 different model sizes as shown in Table 4. The cells are shaded with red and green representing True Positive and True Negative respectively. The categories are labeled as NG (not good/damaged) and OK.

According to the results, it can be concluded that YOLO-v5 large provides the best output its highest accuracy of 99.95% in detecting defects, and on average for 10 cross-validations the accuracy is 99.74% (Table 3). Also, other parameters like Misclassification Rate, True Positive Rate, False Positive Rate, True Negative Rate, and Prevalence which can be noticed that measuring parameters for YOLO-v5 large, is consistent and in all the 10 cross-validations for YOLO-v5 large model, there is not a huge difference between them. Figure 5 shows sample images for True Positive. In the above sample images, it can be seen that the model can detect the defects with confidence.

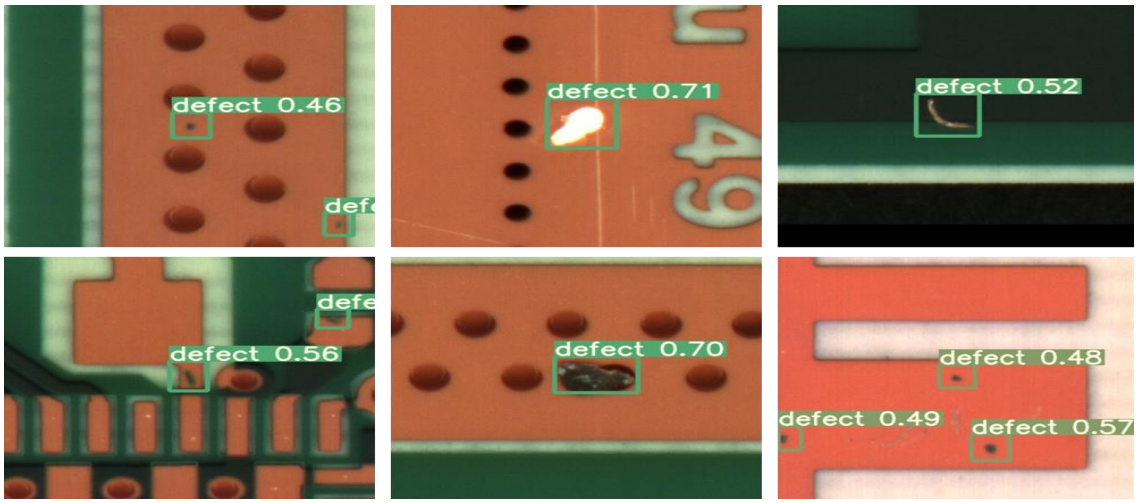


Figure 5. Sample images for True Positive.

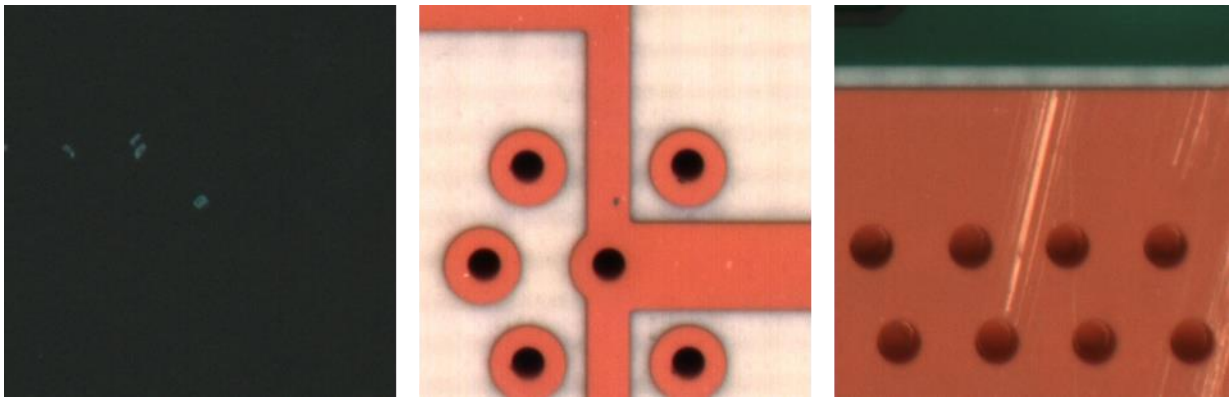


Figure 6. Sample images for False Negative.

Figure 6 displays the sample images for False Negative, and these are the images that have a defective region in them but it's not able to detect it.

Figure 7 shows the False Positive defects. In these images, the model predicts False Positive which is misclassified but it with low confidence. To avoid such kind of misclassification, the model needs to be fined tuned by inspecting the size of the bounding box in training data.

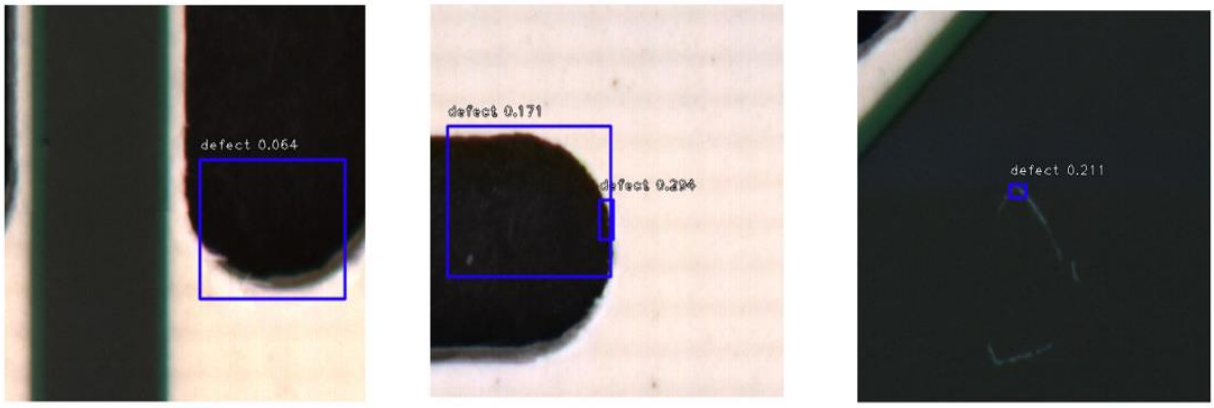


Figure 7. Sample images for False Positive.

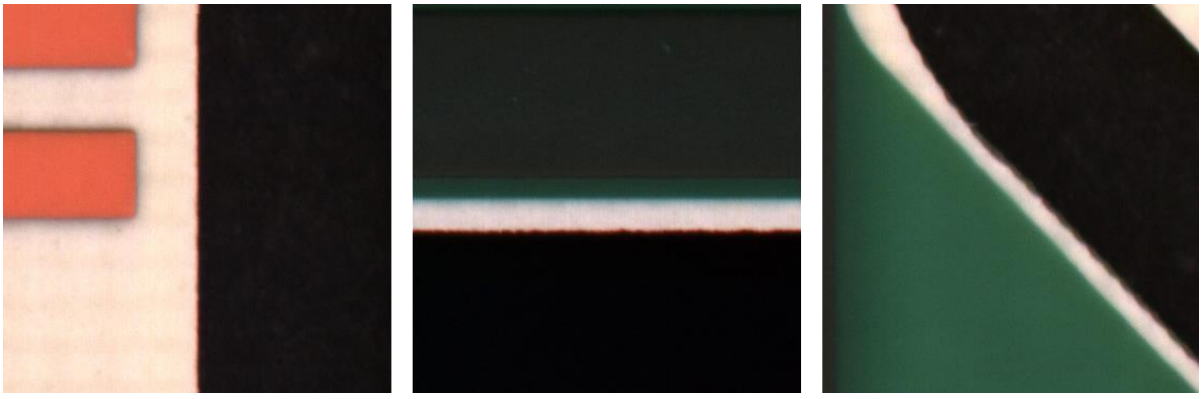


Figure 8. Sample images for True Negative.

Figure 8 displays sample images for True Negative. These samples do not have any defect and have been detected as OK. As it can be seen, the detection results YOLO-v5 large are appreciable with an average accuracy of 99.74% apart from the evaluation precision which is consistently 0.99 (Table 3). In addition, other measures such as misclassification rate, True Positive Rate, False Positive Rate, True Negative Rate, and Prevalence are the best using YOLO-v5 large compared to YOLO-v5 medium and YOLO-v5 small which gives stability and consistency. In most machine learning algorithms, it is believed that a large and balanced dataset makes the difference in performance.

Table 5. Accuracy, misclassification rate, true positive rate, false positive rate, true negative rate, precision, prevalence, mean and standard deviation of accuracy for YOLO-v5 large model.

YOLO-v5 large							
Category	Accuracy	Misclassification Rate	True Positive Rate	False Positive Rate	True Negative Rate	Precision	Prevalence
Cross validation 1	99.60%	0.0039	0.99	0.013	0.98	0.99	0.80
Cross validation 2	99.35%	0.0065	0.99	0.020	0.97	0.99	0.80
Cross validation 3	99.47%	0.0052	0.99	0.006	0.99	0.99	0.80
Cross validation 4	99.47%	0.0052	0.99	0	1	1	0.80
Cross validation 5	99.73%	0.0026	0.99	0.006	0.99	0.99	0.80
Mean ± SD	99.524 ± 0.12						

*Note. Similar to Table 1.

Table 6. Accuracy, misclassification rate, true positive rate, false positive rate, true negative rate, precision, prevalence, mean and standard deviation of accuracy for Tiny-YOLO-v2 model.

Tiny-YOLO-v2 Batch size = 32							
Category	Accuracy	Misclassification Rate	True Positive Rate	False Positive Rate	True Negative Rate	Precision	Prevalence
Cross validation 1	98.82%	0.01	0.99	0.02	0.97	0.99	0.80
Cross validation 2	98.95%	0.01	0.99	0.02	0.97	0.99	0.80
Cross validation 3	98.82%	0.01	0.98	0.01	0.98	0.99	0.80
Cross validation 4	99.21%	0.007	0.99	0.02	0.97	0.99	0.80
Cross validation 5	98.16%	0.01	0.98	0.04	0.95	0.99	0.80
Mean ± SD	98.79 ± 0.346						

*Note. Similar to Table 1.

In order to compare with our previous study using Tiny-YOLO-v2 version [27], five-fold cross-validation with batch size 32 were used in this experiment where Tiny-YOLO-v2 was tested using 765 images. An average defective PCB detection accuracy (batch size of 32) was found to be 98.79%, and evaluation precision was consistently 0.99. In addition, other measures such as the misclassification rate, true positive rate, false-positive rate, true negative rate, and prevalence for a batch size of 32 were

not up to the mark in comparison with YOLO-v5 as deposited in Tables 5 and 6. As the mean accuracy for YOLO-v5 large is 99.52%. Table 7 displays Confusion matrix of five different cross-validations and comparison between Tiny-YOLO-v2 and YOLO-v5 large methods.

Finally, in order to compare the training time and memory size for our proposal YOLO-v5 three models (i.e., small, medium, and large). Table 8 shows results for running 150 epochs.

Table 7. Confusion matrix of five different cross-validations and comparison between Tiny-YOLO-v2 and YOLO-v5 large methods.

Batch size		Yolo-v5(l)		Tiny-YOLO-v2 (batch size 32)	
		NG	OK	NG	OK
Cross validation 1	NG	615	2	611	6
	OK	1	147	3	145
Cross validation 2	NG	614	3	613	4
	OK	2	146	4	144
Cross validation 3	NG	616	1	610	7
	OK	3	145	2	146
Cross validation 4	NG	617	0	614	3
	OK	4	144	3	145
Cross validation 5	NG	616	1	609	8
	OK	1	147	6	142

Table 8: Training time and memory requirements for our proposal YOLO-v5 three models (i.e., small, medium, and large) with 150 epochs.

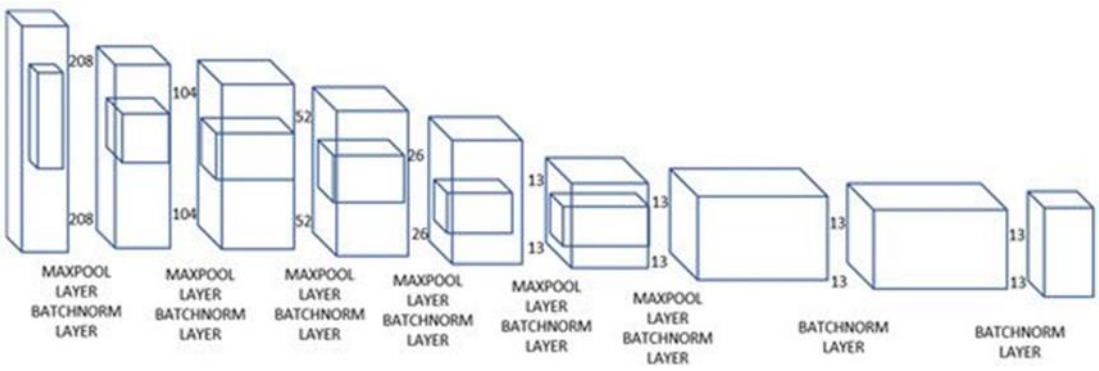
	YOLO-v5 small	YOLO-v5 medium	OLO-v5 large
Time (hrs.)	3 hrs	15 hrs	31 hrs
Memory (MB)	27 MB	50 MB	192 MB

*Note. Time is the required time used by model for training process and memory is amount of memory of weight that is used by model.

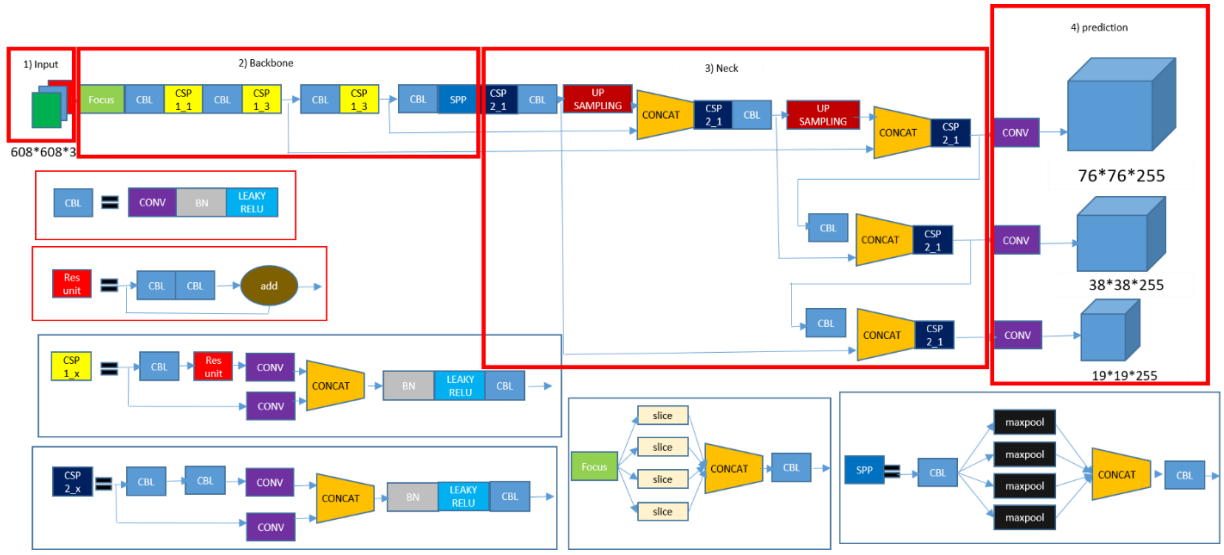
4. Discussion

One major advantage of YOLO-v5 over other models in the YOLO series is that YOLO-v5 is coded in PyTorch from the ground up. This makes it useful for machine learning engineers as there exists an active and vast PyTorch community to support the researchers. YOLO-v5 is also much faster than all the previous versions of YOLO [29–31]. In addition to this, YOLO-v5 is nearly 90% smaller than YOLO-v4. This means YOLO-v5 can be deployed to embedded devices much more easily. To know more about some of the advantages of YOLO-v5, mosaic augmentation, is an included technique in the improved YOLO-v5. Previously, YOLO models are developed using darknet and that was not so flexible for research work and not suitable to be used in industry. Iterating on YOLO-v5 may be easier for the broader research community. Apart from that, YOLO-v5 is fast running on NVidia Titan V as it can reach 140 FPS while the other YOLO models are restricted to 50 FPS. YOLO-v5 is accurate

after training for 1000 epochs and roughly it can achieve 0.935 mean average precision. It is not usually seen in any other YOLO or object detection models without loss an accuracy has been achieved. Finally, the size of YOLO-v5 model, e.g., the weight file of YOLO-v5 small is only 27 megabytes and the size of YOLO-v5 large is 192 megabytes.



(a)



(b)

Figure 9. The comparison of previous Tiny YOLO-v2 and proposal YOLO-v5 models (a) Structure of Tiny-YOLO-v2; (b) Structure of YOLO-v5.

As mentioned in Section 2, adaptive image scaling and Non-maximum suppression are the changes that we have done comparative to our previous work [27]. In this study we have also implemented the automatic testing process and a user interface which automatically extracts the images from AVI machine and runs the testing in a parallel computing which eventually saves time during testing process. In addition to that, it runs the testing without intervention of human. Furthermore, to justify the difference between both the model, Figure 9(a) is the structure of Tiny-YOLO-v2 that we have used in the previous study, and Figure 9(b) displays the structure of YOLO-v5 that we have used

in this research. As we can see in the below figures, the difference between both the structure is the neck section. In the field of target detection, in order to better extract the fusion features, usually some layers are inserted in the Backbone and the output layer. This part is called Neck. The neck, which is equivalent to the target detection network, is also very critical. YOLO-v5 uses CSPDarknet53 as the backbone, plus the SPP module, PANET as the neck, and the head of YOLO-v3. Compared with Tiny-YOLO-v2, the structure diagram of YOLO-v5 has more CSP structure and PAN structure. If you simply look at the structure, you will find it very convoluted. However, after seeing the below structure, it will feel suddenly open. In fact, the overall structure is same, but using various new algorithm ideas to improve each substructure. Yolo-v5 structure is the method for neighboring positive sample anchor matching strategy. Through flexible configuration parameters, models of different complexity can be obtained so it improves overall performance through some built-in hyper-parameter optimization strategies, for example, mosaic enhancement is used to improve the detection performance of small objects.

The used datasets are collected by our research team who have decades of experience in quality inspection of PCB. Moreover, the traditional deep learning [32–35] methods are based on classifying or detecting particular objects in the image. In this paper, three different types of models have been deployed and have been compared with each other. The training time has also been compared each other after comparing training time, it has been observed that YOLO-v5 small takes less time compared to the other two models. YOLO-v5 small takes almost 3–4 hrs., to train and YOLO-v5 medium takes 12–14 hrs. to train while the YOLO-v5 large takes 31–32 hrs. However, with respect to the overall accuracy, YOLO-v5 large is more accurate than the other two models. YOLO-v5 small has an average accuracy of 97.52%, while YOLO-v5 medium has 99.16% and YOLO-v5 large has an accuracy of 99.74%. Apart from accuracy if we compare with the map, YOLO-v5 large has the highest map than the other two. YOLO-v5 large has 94%, while YOLO-v5 medium and YOLO-v5 small has 84 and 82%.

5. Conclusions

This research proves that YOLO-v5 large can detect the defects in PCB with plausible accuracy of 99.74%, which optimized a lot of skilled manpower and time. It also increases the accuracy. In future work, the accuracy can be increased considering several types of defects. In future work, efficient performance with higher accuracy requires further research for example includes a different kind of defects. The grouping of classes must be done in a more balanced way, and we need to include more types of defects and increase the type of defects with an increase in data. Further, in the future, we will try to develop fully automatic training without human interference and use transfer learning and meta-learning to improve the accuracy. Eventually, the transfer learning approach [36,37] can be considered for a pre-trained YOLO model.

Acknowledgments

This work was financially supported by the Ministry of Science and Technology, Taiwan (Grant number: MOST 109-2622-E-155-007).

Conflict of interest

The authors declare no conflict of interest in this paper.

References

1. H. Suzuki, Junkosha Co Ltd., Printed Circuit Board, US 4640866, March 16, 1987.
2. H. Matsubara, M. Itai, K. Kimura, NGK Spark Plug Co Ltd., Printed Circuit Board, US 6573458, September 12, 2003.
3. J. A. Magera, G. J. Dunn, The Printed Circuit Designer's Guide to Flex and Rigid-Flex Fundamentals, Motorola Solutions Inc., Printed Circuit Board, US 7459202, August 21, 2008.
4. H. S. Cho, J. G. Yoo, J. S. Kim, S. H. Kim, Official Gazette of the United states patent and trademark, Samsung Electro Mechanics Co Ltd., Printed Circuit Board, US 8159824, 2012.
5. A. P. S. Chauhan, S. C. Bhardwaj, Detection of bare PCB defects by image subtraction method using machine vision, in *Proceedings of the World Congress on Engineering*, **2** (2011), 6–8.
6. N. K. Khalid, Z. Ibrahim, *An Image Processing Approach towards Classification of Defects on Printed Circuit Board*, PhD thesis, University Technology Malaysia, Johor, Malaysia, 2007.
7. Y. Bengio, A. Courville, P. Vincent, Representation learning: A review and new perspectives, *IEEE Trans. Pattern Anal. Mach. Intell.*, **35** (2013), 1798–828.
8. P. S. Malge, PCB defect detection, classification and localization using mathematical morphology and image processing tools, *Int. J. Comput. Appl.*, **87** (2014), 40–45.
9. Y. Takada, T. Shiina, H. Usami, Y. Iwahori, Defect detection and classification of electronic circuit boards using keypoint extraction and CNN features, in *The Ninth International Conferences on Pervasive Patterns and Applications Defect*, **100** (2017), 113–116.
10. D. B. Anitha, R. Mahesh, A survey on defect detection in bare PCB and assembled PCB using image processing techniques, in *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, (2017), 39–43.
11. A. J. Crispin, V. Rankov, Automated inspection of PCB components using a genetic algorithm template-matching approach, *Int. J. Adv. Manuf. Technol.*, **35** (2007), 293–300.
12. F. Raihan, W. Ce, PCB defect detection USING OPENCV with image subtraction method, in *2017 International Conference on Information Management and Technology (ICIMTech)*, (2017), 204–209.
13. I. B. Basyigit, A. Genc, H. Dogan, F. A. Senel, S. Helhel, Deep learning for both broadband prediction of the radiated emission from heatsinks and heatsink optimization, *Eng. Sci. Technol. Int. J.*, **24** (2021), 706–714.
14. S. Metlek, K. Kayaalp, I. B. Basyigit, A. Genc, H. Doğan, The dielectric properties prediction of the vegetation depending on the moisture content using the deep neural network model, *Int. J. RF Microwave Comput.-Aided Eng.*, **31** (2020), e22496.
15. H. Hosseini, B. Xiao, M. Jaiswal, R. Poovendran, On the limitation of convolutional neural networks in recognizing negative images, in *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, (2017), 352–358.
16. X. Tao, Z. Wang, Z. Zhang, D. Zhang, D. Xu, X. Gong, et al., Wire defect recognition of spring-wire socket using multitask convolutional neural networks, *IEEE Trans. Compon. Package. Manuf. Technol.*, **8** (2018), 689–698.

17. J. R. R. Uijlings, K. E. A. Van De Sande, T. Gevers, A. W. M. Smeulders, Selective search for object recognition, *Int. J. Comput. Vision*, **104** (2012), 154–171.
18. R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, (2014), 580–587.
19. K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, preprint, arXiv:1512.03385.
20. K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, preprint, arXiv:1409.1556.
21. C. Szegedy, S. Ioffe, V. Vanhoucke, Inception-v4, inception-resnet and the impact of residual connections on learning, preprint, arXiv:1602.07261.
22. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, et al., Going deeper with convolutions, preprint, arXiv:1409.4842.
23. N. Suda, V. Chandra, G. Dasika, A. Mohanty, Y. Ma, S. Vrudhula, et al., Throughput-optimized OpenCL-based FPGA accelerator for large-scale convolutional neural networks, in *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, (2016), 16–25.
24. J. Zhang, J. Li, Improving the performance of OpenCL-based FPGA accelerator for convolutional neural network, in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, (2017), 25–34.
25. D. Wang, J. An, K. Xu, PipeCNN: An OpenCL-based FPGA accelerator for large-scale convolution neuron networks, preprint, arXiv:1611.02450.
26. J. Cong, B. Xiao, Minimizing computation in convolutional neural networks, in *International conference on artificial neural networks*, Springer, Cham, (2014), 281–290.
27. V. A. Adibhatla, H. C. Chih, C. C. Hsu, J. Cheng, M. F. Abbod, J. S. Shieh, Defect detection in printed circuit boards using you-only-look-once convolutional neural networks, *Electronics*, **9** (2020), 1547.
28. M. Pritt, G. Chern, Satellite image classification with deep learning, in *2017 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, (2017), 1–7.
29. X. S. Zhang, R. J. Roy, E. W. Jensen, EEG complexity as a measure of depth of anesthesia for patients, *IEEE Trans. Biomed. Eng.*, **48** (2001), 1424–1433.
30. V. Lalitha, C. Eswaran, Automated detection of anesthetic depth levels using chaotic features with artificial neural networks, *J. Med. Syst.*, **31** (2007), 445–452.
31. M. Peker, B. Sen, H. Gürüler, Rapid automated classification of anesthetic depth levels using GPU based parallelization of neural networks, *J. Med. Syst.*, **39** (2015), 1–11.
32. P. L. Callet, C. Viard-Gaudin, D. Barba, A convolutional neural network approach for objective video quality assessment, *IEEE Trans. Neural Networks*, **17** (2006), 1316–1327.
33. D. C. Cireşan, U. Meier, L. M. Gambardella, J. Schmidhuber, Convolutional neural network committees for handwritten character classification, in *Proceedings of the 2011 International Conference on Document Analysis and Recognition*, (2011), 1135–1139.
34. N. Kalchbrenner, E. Grefenstette, P. Blunsom, A convolutional neural network for modelling sentences, preprint, arXiv:1404.2188.
35. A. Devarakonda, M. Naumov, M. Garland, Adabatch: Adaptive batch sizes for training deep neural networks, preprint, arXiv:1712.02029.

36. T. Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollár, Focal loss for dense object detection, in *Proceedings of the IEEE international conference on computer vision*, (2017), 2980–2988.
37. L. Shao, F. Zhu, X. Li, Transfer learning for visual categorization: A survey, *IEEE Trans. Neural Netw. Learn. Syst.*, **26** (2015), 1019–1034.

AIMS Press