electronics

MDPI

Article

Defect Detection in Printed Circuit Boards Using You-Only-Look-Once Convolutional Neural Networks

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

1 Department of Mechanical Engineering, Yuan Ze University, Taoyuan 32003, Taiwan; venkataniladibhatla@gmail.com (V.A.A.); jsshieh@saturn.yzu.edu.tw (J.-S.S.)
2 Department of Advanced Manufacturing System, Boardtek Electronics Corporation, Taoyuan 328454, Taiwan; strong@mail.boardtek.com.tw (H.-C.C.); chang@mail.boardtek.com.tw (C.-C.H.); JosephCheng@mail.boardtek.com.tw (J.C.)
3 Department of Electronic and Computer Engineering, Brunel University London, Uxbridge UB8 3PH, UK
* Correspondence: maysam.abbod@brunel.ac.uk

Received: 31 July 2020; Accepted: 18 September 2020; Published: 22 September 2020

Abstract: In this study, a deep learning algorithm based on the you-only-look-once (YOLO) approach is proposed for the quality inspection of printed circuit boards (PCBs). The high accuracy and efficiency of deep learning algorithms has resulted in their increased adoption in every field. Similarly, accurate detection of defects in PCBs by using deep learning algorithms, such as convolutional neural networks (CNNs), has garnered considerable attention. In the proposed method, highly skilled quality inspection engineers first use an interface to record and label defective PCBs. The data are then used to train a YOLO/CNN model to detect defects in PCBs. In this study, 11,000 images and a network of 24 convolutional layers and 2 fully connected layers were used. The proposed model achieved a defect detection accuracy of 98.79% in PCBs with a batch size of 32.

Keywords: convolution neural network; YOLO; deep learning; printed circuit board

1. Introduction

Printed circuit boards (PCBs) are a crucial component in most electronic devices. For decades, PCBs have been adapted for industrial and domestic use. PCBs are the primary component of any electronic design and have been used in fields such as logistics, defence, and aeronautics as well as for applications in the automobile and medical industries, among others. PCBs are solid thin plates prepared from laminated materials, fibreglass, or composite epoxy and form a physical base that supports chips and electronic components [1–4]. These boards are designed with conductive pathways, which form circuits and power electronic devices attached to the PCBs. Therefore, PCB inspection processes have continually improved to meet the ever-increasing demands of modern manufacturing. Production and manufacturing industries have attempted to achieve 100% quality assurance for all PCBs. Automated visual inspection of PCBs has advanced considerably [5,6]. Studies [7] have revealed that deep learning outperforms traditional machine-based classification and feature extraction algorithms. Defect detection in PCBs during quality inspection is critical. In the conventional method, defects are initially detected by an automatic inspection (AOI) machine. A skilled quality inspection engineer then verifies each PCB. Many boards classified as defective by the AOI machine may not be defective. The machine can erroneously classify a PCB as defective because of a scratch or small hole or the presence of nanoparticles such as dust, paper fragments, or small air bubbles. Slight variation from the reference sample may result in the AOI machine classifying PCBs as defective. However, reliance on

inspection engineers requires considerable human resources as well as sufficient training. Furthermore, even skilled operators can make errors during inspection. Therefore, robust deep learning systems can replace skilled operators to a certain extent. Machines programmed with deep learning algorithms can be used to verify defects. Deep learning methods are faster and more accurate than skilled operators are. Thus, the objective of this study was to develop a deep learning-based PCB recognition system that reduces the false detection rate and increases the production rate.

With the increasing popularity of consumer electronics products, such as laptops, smartphones, and tablets, accurate PCB manufacturing is critical [8]. Because of this surge in the demand for PCBs in the market, manufacturers are required to produce PCBs in large quantities. Therefore, maintaining the quality of such large numbers of PCBs is challenging. Accurate automated inspection systems can prove helpful in quality maintenance. Such systems overcome the limitations of manual inspection for a large number of PCBs. Automated visual PCB inspection can provide fast and quantitative information of defects and therefore can prove to be an asset in manufacturing processes. A few examples of PCB defect detection methods can be found in the literature [8–10]. Typically, the template-matching method [11] is used to detect defects in PCBs. Another method for PCB defect detection is OPENCV followed by image subtraction [12]. However, these detection algorithms are limited to a specific type of defect in PCBs. Remarkable progress has been made in the use of convolutional neural networks (CNNs) in several applications, such as image recognition [13,14] and object detection. In particular, object detection is achieved by implementing object recognition methods [15] and region-based CNNs [16]. In this study, a CNN classifier was trained to recognise various electrical components on a PCB and then detect and localise defects on the PCB components by using Tiny-YOLO-v2, an effective and accurate object detector.

Quality inspection engineers are employed to ensure minimum defects. Manual inspection and diagnosis of defects is challenging. Defect inspection encompasses detection of several types of defects, an extremely low tolerance for errors, and considerable expertise to reliably recognise and handle flawed units.

Researchers have applied various you-only-look-once (YOLO) approaches to art data sets and achieved excellent accuracy. YOLO, an object detection algorithm, differs from other classification methods. Unlike conventional classification mechanisms, YOLO can classify more than one object in a single picture. Because YOLO is based on CNNs, it has a unique method of object detection. In the algorithm, a single neural network is used for an image separated into different regions. The probability of each region is then predicted, and predicted probabilities determine the location of bounding boxes.

YOLO is becoming increasingly popular because of its high accuracy and efficiency. As the name suggests, an image is checked just once; that is, a single forward propagation pass is performed by the neural network to generate predictions. Following the nonmax suppression technique, the algorithm outputs the recognised objects using the bounding box. The YOLO algorithm with a single CNN predicts several bounding boxes and the class probability for those boxes. This algorithm uses full images for training and optimises its detection performance. YOLO is quick and considers the whole image during processing. Therefore, it encrypts circumstantial details regarding class. The YOLO algorithm is faster than other object detection algorithms because it learns general representations of objects during training.

However, the YOLO algorithm has some drawbacks, such as more localisation errors than Fast R-CNN. Furthermore, YOLO has a lower recall than other region proposal–based methods. Thus, the objective of this study was to improve recall and localisation while maintaining high classification accuracy. Currently, deep networks are preferred for computer vision [17–19]. Excellent results can be achieved using large networks and different methods. YOLO is preferred because of its accurate detection within a limited time frame. The performance of YOLO has been continually improved over time.

Networks that can achieve fast and precise detection are always preferred. Several applications, such as self-driving cars and robotics, require low latency prediction. Thus, YOLO-v2 is designed to

be fast. Most object detection or classification frameworks use VGG-16 as the basic feature extractor algorithm [19] because of its high accuracy and robustness. However, VGG-16 is highly complex and requires 30.69 billion floating point operations per second (FLOP) for a single pass over to achieve an image resolution of 224 × 224. A custom network based on the Google Net architecture is typically preferable for YOLO frameworks [20]. This model is faster than VGG-16 and requires only 8.52 billion FLOP for a forward pass. However, its accuracy is slightly lower than that of VGG-16. Therefore, on the basis of these parameter comparisons, we used Tiny-YOLO-v2 (a modified and compact version of YOLO).

Using Microsoft visual studio, we developed an interface to collect images from the AOI machine. The interface enables the quality inspection engineer to label the defective region on individual PCBs. Deep learning is then applied on the gathered image data. This study implemented the Tiny-YOLO-v2 model with a CNN to improve the accuracy and reduce the error rate.

2. Materials and Methods

2.1. PCB Data Set

The PCB data set was obtained from the AOI machine to generate an image of the reference PCB sample in RGB format, which was then converted into JPEG format and saved. In addition to images of the reference sample, images of defective samples were also collected using the AOI machine. The images were extracted and cropped. Almost 11,000 images were collected for training. An interface was developed (as shown in Figure 1) to enable quality inspection engineers to label the defective regions of PCBs, which were then compared with the same location on the reference sample.

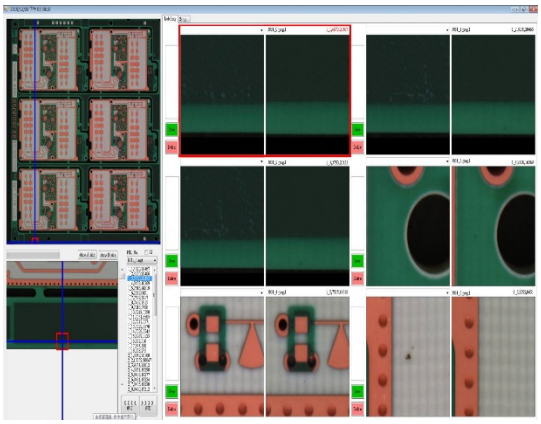
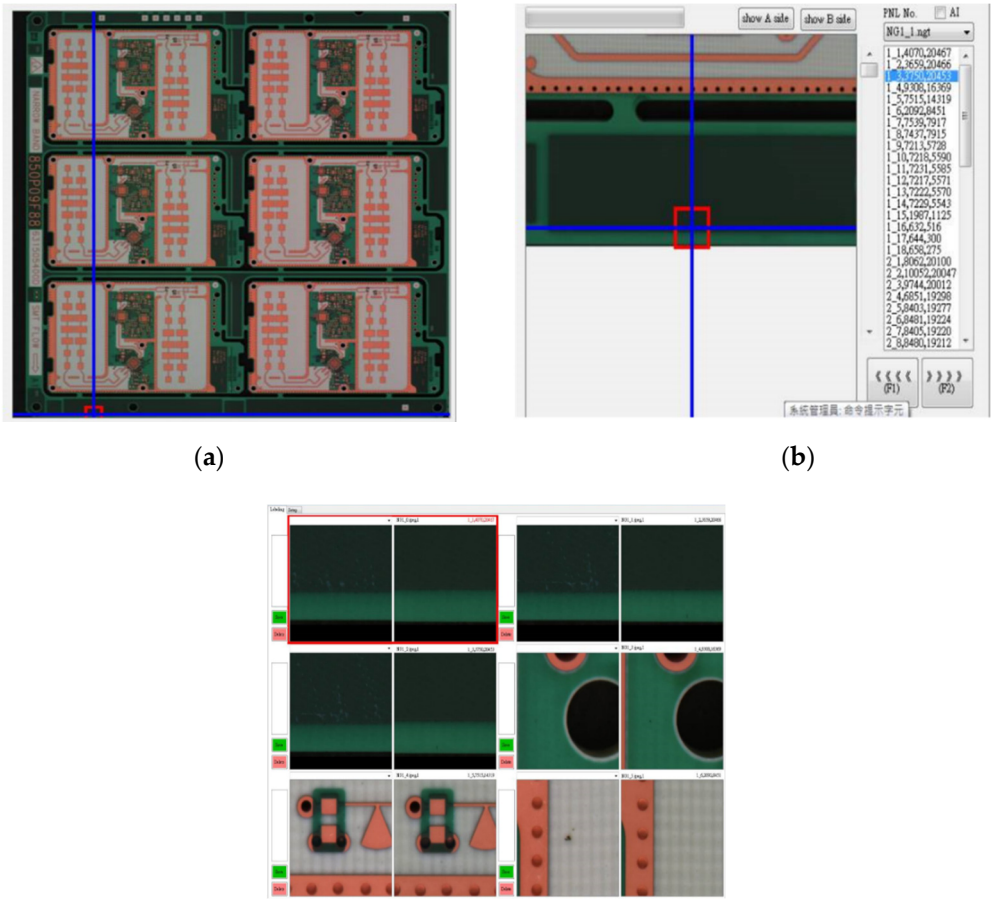


Figure 1. User interface for data collection.

Figure 2 illustrates this process. Figure 2a displays six identical PCBs with a defect at the bottom. Figure 2b is an enlarged image of the defective region, and Figure 2c displays the defective region and the same region on the reference sample. The images were 420 × 420 pixels and were placed adjacent to each other to enable a quality inspection engineer to compare the defective and reference samples. The inspection engineer then labelled the defective area in the image. Images of 11 defect types were collected. However, because of the small amount of data, all the defect types were labelled as a single type of defect.



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

(**c**)

Figure 2. Sections of the user interface. (**a**) Printed circuit board (PCB) panels. (**b**) Cropped or enlarged section of the defective region. (**c**) Cropped or enlarged area of the reference sample.

2.2. *Architecture of Tiny-YOLO-v2*

The unique object detection algorithm [21] developed using Tiny-YOLO-v2 [22,23] is explained in this section. For a single image, Tiny-YOLO-v2 predicts several bounding boxes with the class probability by using a single CNN. Figure 3 displays the structure of Tiny-YOLO-v2, which includes convolutional layers, various activation function blocks—such as a rectified linear unit (ReLU) and batch normalisation, within the region layer—and six max pooling layers. An image of 416 × 416 pixels is used as the input for the classifier. The output is a (125 × 13 × 13) tensor with 13 × 13 grid cells. Each grid cell corresponds to 125 channels, consisting of five bounding boxes predicted by the grid cell and the 25 data elements that describe each bounding box (5 × 25 = 125).

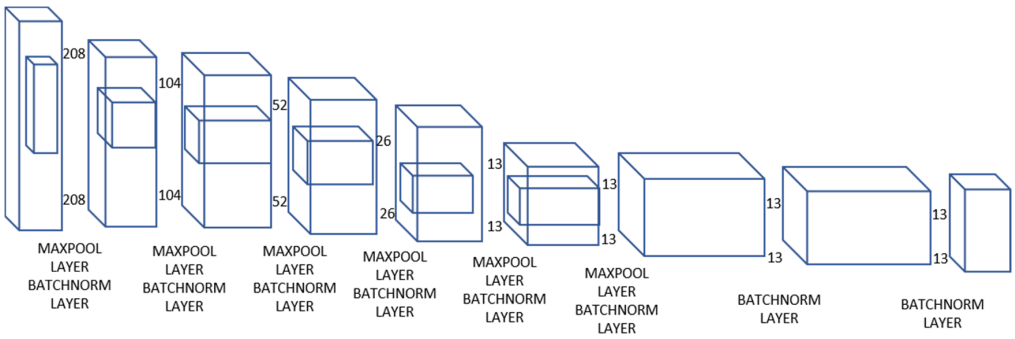


Figure 3. Tiny-YOLO (you-only-look-once)-v2 Architecture.

2.3. Convolutional Layer

The convolutional layer in Tiny-YOLO-v2 occupies 90% of the feed-forward computation time [24]. Therefore, the performance of the classifier is improved through the optimisation of the convolutional layer. Hundreds of millions of addition and multiplication operations are performed between the local regions and filters for a single image. The function is presented as follows:

$$X^{(i)} = \sum_{i=1}^{n} \left(X^{(j)} \times W^{(k)} \right) + b \tag{1}$$

where $X^{(i)}$ = output pixel feature, $X^{(j)}$ = input pixel feature, $W^{(k)}$ = convolution weights, and b = convolution bias.

The number of functions involved in the convolution layer is calculated according to Equation (2). The number of operations for batch normalisation and the leaky activation function for each layer are ignored in this equation.

$$Operations = 2 \times N_{im} \times K \times K \times N_{out} \times H_{out} \times W_{out} \tag{2}$$

where N_{im} = the number of channels of the input feature, K = the filter size; N_{out} = the number of filters; H_{out} = the output feature height; W_{out} = the output feature width. The required memory is a challenge because of the paucity of space. The weight used in the convolutional layer is the primary parameter in the Tiny-YOLO-v2. Equation (3) expresses the weights involved in the convolutional layer:

$$Weights = N_{in} \times K \times K \times N_{out} \tag{3}$$

where N_{in} = the number of channels for the input feature, K = the filter size, and N_{out} = the number of filters. Approximately 7 billion operations and 15 million weights are simultaneously inputted in Tiny-YOLO-v2 for an input image in Pascal VOC. Furthermore, 5.7 billion operations with 12 million weights are inputted in Tiny-YOLO-v2 for a single input image in the COCO data set.

2.4. Activation Function

In the CNN architecture, the activation function is used to correct the input. Sigmoidal activation functions are the most used activation function and are limited to the maximum and minimum values; thus, they lead saturated neurons to higher layers of the neural network. The leaky ReLU function updates weights, which are never reactivated on any data point, as shown in Figure 4.

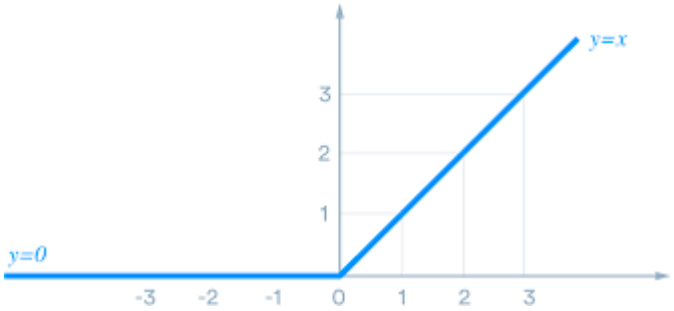


Figure 4. Leaky rectified linear unit.

2.5. Pooling Layer

The pooling layer is used to reduce the dimensions of images. The main objective of the pooling layer is to eliminate unnecessary information and preserve only vital parameters. Often used

are maximum pooling, which considers the maximum value from the input, and average pooling, which considers the average value, expressed as follows:

$$s[i, j] = \max \{S[i;, j']:i \leq i' < i + ,j \leq j' < j + p\} \quad (4)$$

$$s[i, j] = \text{average } \{S[i;, j']:i \leq i' < i + ,j \leq j' < j + p\} \quad (5)$$

Layers in Tiny-YOLO-v2 can be accessed after the application of the batch normalisation layer to convolutional layers. Inputs with zero mean and unit variance are used. Batch normalisation is expressed in Equation (6). The earlier output of the convolutional layer is normalised through removal of the batch mean and dissection of the batch variance. The output after batch normalisation is shifted according to bias and scaled. CNNs consist of variables such as variance, mean, scale, and bias caused during the CNN training stage. These parameters permit individual layers to learn independently and prevent overfitting through their regularisation effect.

$$x^{(j)} = \frac{(x^{(i)} - \mu)}{\sqrt{\sigma^2} + \xi} \quad (6)$$

where $x^{(j)}$ = the output pixel after batch normalisation, $x^{(i)}$ = the output pixel after convolution, μ = the mean, σ = variance, ξ = constant.

3. Results

Efficient computing is necessary to achieve optimal performance of the Tiny-YOLO-v2 model. Therefore, a NVidia TITAN V graphics processing unit (GPU) (NVidia, Santa Clara, CA, USA) was used for this experiment, which reduced the training time to 10% (i.e., 34 to 4 h). The Tiny-YOLO-v2 model was trained with the Keras framework running on the NVidia TITAN V GPU by using the Linux operating system. The early stop criteria were implemented, which achieved the highest validation accuracy. A batch size of 32 was used, which is a standard maximum batch size. Here, 8 GB of memory was used. Initially, during the training stage, a small data set was selected for testing the basic performance of these networks. The network settings and parameters were adjusted and tuned gradually using the trial-and-error method. The parameter batch size was changed. Initially, 5000 images of PCBs labelled as defective were used. Next, 11,000 images of defective PCBs were used. This procedure was implemented to improve the accuracy of the Tiny-YOLO-v2 model and regulate the parameters of the model to attain the most advantageous implementation of the training model. The epoch size is based on the training data set. After parameters were selected for the model, an initial ideal start for training was initiated. Moreover, with the callback function, a model checkpoint instruction was used to tune the training process. Its primary purpose is to save the Tiny-YOLO-v2 model with all the weight after each epoch so that finally model framework and weights can save its optimal performance.

Fivefold cross validation [25] was used to evaluate the execution of the trained models. Initially, the data were randomly divided into five equal segments, four of which were used for the training model, and the remaining segment was used for testing. After every training phase, the model evaluated the remaining segment. This procedure was performed five times with different testing and training data sets. After completing the training process, each model was tested on a different data segment. Figure 5 displays the confusion matrix of the tests. The accuracy of the batch sizes of 8, 16, and 32 was approximately 95.88%, 97.77%, and 98.79%, respectively. Cross validation training was performed for batch sizes of 8, 16, and 32. Fifteen cross validations were performed on three batch sizes (Figure 5). Green cells represent true positives and true negatives, and red cells represent false positives and false negatives. The cells are labelled as NG (not good/damaged) or OK.

		Real					
Batch size		**8**		**16**		**32**	
		NG	OK	NG	OK	NG	OK
Cross validation 1	NG	592	25	607	10	611	6
	OK	8	140	3	145	3	145
Cross validation 2	NG	590	27	604	13	613	4
	OK	6	142	4	144	4	144
Cross validation 3	NG	595	22	608	9	610	7
	OK	4	144	8	140	2	146
Cross validation 4	NG	591	26	606	9	614	3
	OK	9	139	2	146	3	145
Cross validation 5	NG	594	23	605	12	609	8
	OK	7	141	9	139	6	142

Predicted

Figure 5. Confusion matrix of five different cross validations for batch size 8, 16 and 32. Note. True positives (TP): PCB predicted as damaged (NG) and is damaged (NG). True negatives (TN): PCB predicted as OK and does not have any damage (OK). False positives (FP): PCB predicted damaged (NG) but does not have any damage. False negatives (FN): PCB predicted as OK but is damaged (NG).

The results gradually improved as the batch size increased, which proved that the Tiny-YOLO-v2 model is more efficient than other CNN models. After every epoch or iteration, the accuracy of the training process increased, which eventually improved the performance of the model. The final model was saved after the accuracy stabilised. The results of fivefold cross validation with batch sizes of 8, 16 and 32 are displayed in Tables 1–3, respectively.

Figure 6 displays a sample detected as false negative. The PCB was defective and incorrectly classified. Data of 11 types of defects were collected and labelled. The number of images for each defect type was not equal. Therefore, the displayed sample images display the types of defects that were noted least often, and the remaining types of defects are compared.

Figure 7 displays images of a false positive case. The model misclassified the sample and exhibited low confidence. To avoid such misclassification, the size of the bounding box in the training data should be examined.

Figure 8 displays sample images for true positive detections in which defects were detected by the model with confidence.

Figure 9 displays sample images for true negative detection. These samples did not have any defect and were classified as OK. The model achieved accurate defect detection. The average accuracy in detecting defective PCBs for a batch size of 32 was 98.79%, and evaluation precision was consistently 0.99 (Table 3). In addition, other parameters such as the misclassification rate, true positive rate, false positive rate, true negative rate, and prevalence for a batch size of 32 were favourable to those for batch sizes of 8 and 16. In most machine learning algorithms, a large and balanced data set is crucial for achieving optimal performance.

Table 1. Accuracy, misclassification rate, true positive rate, false positive rate, true negative rate, precision, prevalence, and mean and standard deviation of accuracy for a batch size of 8.

Batch Size = 8							
Category	**Accuracy**	**Misclassification Rate**	**True Positive Rate**	**False Positive Rate**	**True Negative Rate**	**Precision**	**Prevalence**
Cross validation 1	95.68%	0.04	0.95	0.05	0.94	0.98	0.80
Cross validation 2	95.67%	0.04	0.95	0.04	0.95	0.98	0.80
Cross validation 3	96.60%	0.03	0.96	0.02	0.97	0.99	0.80
Cross validation 4	95.42%	0.04	0.95	0.06	0.93	0.98	0.80
Cross validation 5	96.07%	0.03	0.96	0.09	0.95	0.98	0.80
Mean ± SD	95.88 ± 0.412%	-	-	-	-	-	-

Note. Accuracy: how often the detection is correct (TP + TN)/total. Misclassification rate: how often detection is incorrect (FP + FN)/total. True positive rate: when defects are accurately detected, also known as 'sensitivity' or 'recall'. False positive rate: when defects are detected but are not actually present (FP/actual no). True negative rate: when no defects are detected where no defects are present, also known as 'specificity' (TN/actual no). Precision: positive predictions (TP/predicted yes). Prevalence: how often defects are accurately detected in the samples (actual yes/total).

Table 2. Accuracy, misclassification rate, true positive rate, false positive rate, true negative rate, precision, prevalence, and mean and standard deviation of accuracy for a batch size of 16.

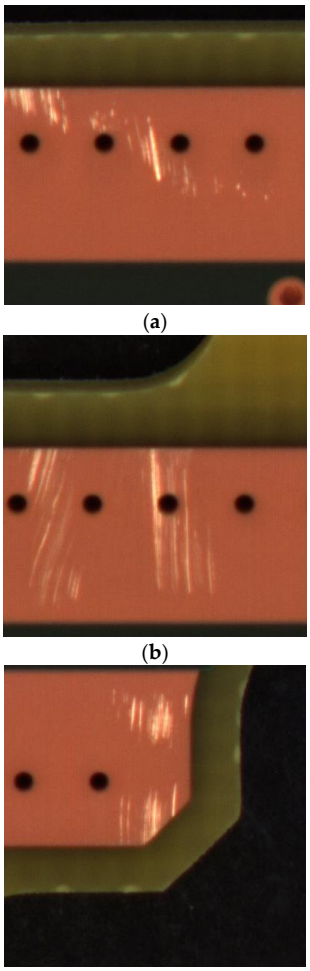
Batch Size = 16							
Category	**Accuracy**	**Misclassification Rate**	**True Positive Rate**	**False Positive Rate**	**True Negative Rate**	**Precision**	**Prevalence**
Cross validation 1	97.77%	0.01	0.98	0.02	0.97	0.99	0.80
Cross validation 2	97.78%	0.02	0.97	0.02	0.97	0.99	0.80
Cross validation 3	97.77%	0.02	0.98	0.05	0.94	0.98	0.80
Cross validation 4	98.30%	0.01	0.98	0.01	0.98	0.99	0.80
Cross validation 5	97.25%	0.02	0.98	0.06	0.93	0.98	0.80
Mean ± SD	97.77 ± 0.32%	-	-	-	-	-	-

Note. Accuracy: how often the detection is correct (TP + TN)/total. Misclassification rate: how often detection is incorrect (FP + FN)/total. True positive rate: when defects are accurately detected, also known as 'sensitivity' or 'recall'. False positive rate: when defects are detected but are not actually present (FP/actual no). True negative rate: when no defects are detected where no defects are present, also known as 'specificity' (TN/actual no). Precision: positive predictions (TP/predicted yes). Prevalence: how often defects are accurately detected in the samples (actual yes/total).

Table 3. Accuracy, misclassification rate, true positive rate, false positive rate, true negative rate, precision, prevalence, mean and standard deviation of accuracy for a batch size of 32.

Batch Size = 32							
Category	**Accuracy**	**Misclassification Rate**	**True Positive Rate**	**False Positive Rate**	**True Negative Rate**	**Precision**	**Prevalence**
Crossvalidation1	98.82%	0.01	0.99	0.02	0.97	0.99	0.80
Crossvalidation2	98.95%	0.01	0.99	0.02	0.97	0.99	0.80
Crossvalidation3	98.82%	0.01	0.98	0.01	0.98	0.99	0.80
Crossvalidation4	99.21%	0.007	0.99	0.02	0.97	0.99	0.80
Crossvalidation5	98.16%	0.01	0.98	0.04	0.95	0.99	0.80
Mean ± SD	98.79 ± 0.346%	-	-	-	-	-	-

Note. Accuracy: how often the detection is correct (TP + TN)/total. Misclassification rate: how often detection is incorrect (FP + FN)/total. True positive rate: when defects are accurately detected, also known as 'sensitivity' or 'recall'. False positive rate: when defects are detected but are not actually present (FP/actual no). True negative rate: when no defects are detected where no defects are present, also known as 'specificity' (TN/actual no). Precision: positive predictions (TP/predicted yes). Prevalence: how often defects are accurately detected in the samples (actual yes/total).

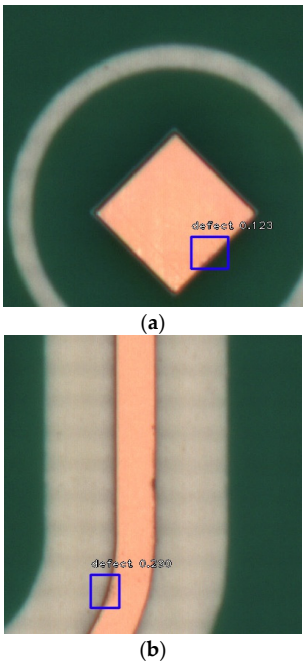


(**a**)

(**b**)

(**c**)

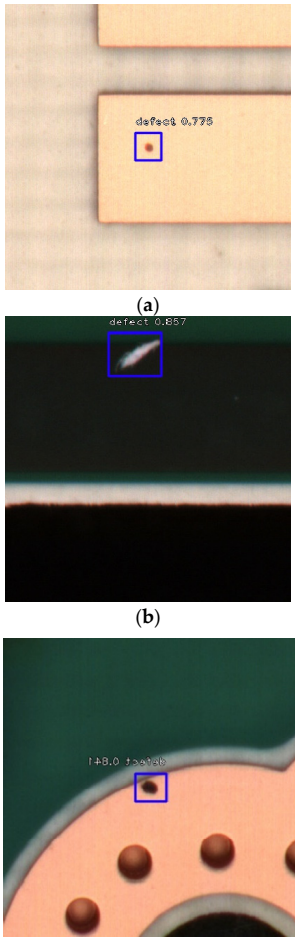
Figure 6. False negative PCB sections: (**a**) False negative sample 1. (**b**) False negative sample 2. (**c**) False negative sample 3.



(**a**)

(**b**)

Figure 7. False positive PCB sections: (**a**) False positive PCB sample 1. (**b**) False positive PCB sample 2.

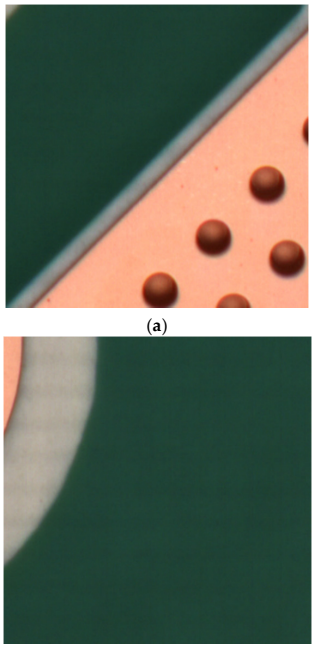


(**a**)

(**b**)

(**c**)

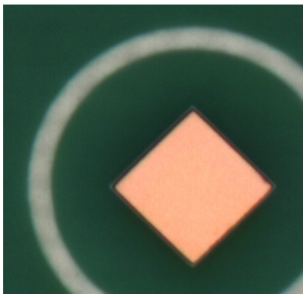
Figure 8. True positive PCB sections: (**a**) True positive PCB sample 1. (**b**) True positive PCB sample 2. (**c**) True positive PCB sample 3.



(**a**)

(**b**)

Figure 9. *Cont*.



(**c**)

Figure 9. True negative PCB sections: (**a**) True negative PCB sample 1. (**b**) True negative PCB sample 2. (**c**) True negative PCB sample 3.

A vanilla version of the CNN [26] was used to compare the results of the proposed model. The vanilla CNN was trained using 15,823 images. Fivefold cross validation was implemented to evaluate the execution of the trained models. Figure 10 displays the results of testing in the form of a confusion matrix (Figure 5); green cells represent true positives and true negatives, and red cells represent false positives and false negatives.

		Real	
		NG	OK
prediction	Cross validation 1	NG	742	180
		OK	90	427
	Cross validation 2	NG	714	106
		OK	118	501
	Cross validation 3	NG	721	191
		OK	111	416
	Cross validation 4	NG	696	156
		OK	136	451
	Cross validation 5	NG	726	136
		OK	106	471

Figure 10. Vanilla convolutional neural network (CNN) confusion matrix of five cross validations. **Note**. True positive (TP): Cases in which damage is detected (NG) in the PCB and the PCB is defective (NG). True negative (TN): the PCB is predicted as OK and does not have any damage (OK). False positive (FP): damage is detected (NG) but the PCB does not actually have any damage. False negative (FN): the PCB is predicted as OK but is actually damaged (NG).

The cross validation results of the Vanilla CNN are presented in Table 4. The mean and standard deviation of accuracy was approximately 81.53 ± 2.326%, and precision was less than 0.8, which is less than that of Tiny-YOLO-v2. The vanilla CNN is an image classifier that could not identify the location of defects. Furthermore, it considers the background as a class, which increased misclassification, unlike Tiny-YOLO-v2, which detects multiple defects in a single image and locates the defects with a bounding box.

Table 4. Accuracy, misclassification rate, true positive rate, false positive rate, true negative rate, precision, prevalence, and mean and standard deviation of accuracy.

Category	Accuracy	Misclassification Rate	True Positive Rate	False Positive Rate	True Negative Rate	Precision	Prevalence
Cross validation 1	81.21%	0.18	0.89	0.29	0.70	0.80	0.57
Cross validation 2	84.55%	0.15	0.85	0.17	0.82	0.77	0.57
Cross validation 3	79.01%	0.20	0.86	0.31	0.68	0.78	0.57
Cross validation 4	79.70%	0.20	0.83	0.25	0.74	0.75	0.57
Cross validation 5	83.18%	0.16	0.87	0.22	0.77	0.78	0.57
Mean ± SD	81.53 ± 2.326%	-	-	-	-	-	-

Note. Accuracy: how often the detection is correct (TP + TN)/total. Misclassification rate: how often detection is incorrect (FP + FN)/total. True positive rate: when defects are accurately detected, also known as 'sensitivity' or 'recall'. False positive rate: when defects are detected but are not actually present (FP/actual no). True negative rate: when no defects are detected where no defects are present, also known as 'specificity' (TN/actual no). Precision: positive predictions (TP/predicted yes). Prevalence: how often defects are accurately detected in the samples (actual yes/total).

4. Discussion

A novel user interface was developed to collect data. Skilled engineers labelled the defects using the interface. The simple method achieved an excellent PCB defect detection accuracy of 98.79%, which is considerably better than that of other algorithms involving complex feature extraction [27–29]. The effectiveness of the proposed method was investigated using various CNN layers. To avoid the delay resulting from the nonspecificity of a single CNN model and insufficient storage capacity, a GPU environment was established. The model was trained using different batch sizes to improve accuracy.

The YOLO strategy is a powerful and quick approach that achieves a higher FPS rate than computationally expensive two-stage detectors (e.g., faster R-CNN) and some single-stage detectors (e.g., RetinaNet and SSD). Tiny-YOLO-v2 was used in this study to increase the execution speed because it is approximately 442% faster than the standard YOLO model, achieving 244 FLOP on a single GPU. A small model size (<50 MB) and fast inference renders the Tiny-YOLO-v2 naturally suitable for embedded computer vision.

Compared with other simple classifiers, YOLO is widely used in practice. It is a simple unified object detection model and can be trained directly using full images. Unlike classifier-based approaches, YOLO is trained using a loss function that directly influences detection performance. Furthermore, the entire model is trained at one time. Fast YOLO is the fastest general-purpose object detector. YOLO-v2 provides the best tradeoff between real-time speed and accuracy for object detection compared with other detection systems across various detection data sets.

Multilayer neural networks or tree-based algorithms are deemed insufficient for modern advanced computer vision tasks. The disadvantage of fully connected layers, in which each perceptron is connected to every other perceptron, is that the number of parameters can increase considerably, which results in redundancy in such high dimensions, rendering the system inefficient. Another disadvantage is that spatial information is disregarded because flattened vectors are used as inputs. However, the key difference distinguishing YOLO from other methods is that the complete image is viewed at one time rather than only a generated region, and this contextual information helps to avoid false positives.

In this study, a modified YOLO model was developed and combined with an advanced CNN architecture. However, some challenges remain to be overcome. Although the proposed system incorporates a smart structure and an advanced CNN method, the model accuracy is not perfect, particularly when operated with unbalanced data sets. The used data sets are collected by experienced PCB quality inspection teams. Moreover, traditional deep learning methods [30–32] are based on classifying or detecting particular objects in an image. Therefore, the model structure was tested with three batch sizes. As elaborated elsewhere [33], these structures exhibit high precision for a classical CNN model but unpredictable performance for the PCB data set. According to research on deep learning, the types of network layer parameters, linear unit activation parameters, regularisation strategies, optimisation algorithms, and approach to batch normalisation of the CNN training process should be focused on for improving the PCB defect detection performance of CNNs. As depicted in Figure 5, the proposed model can accurately detect defects on PCBs and can therefore be used for PCB quality inspection on a commercial scale.

The three models achieved excellent results with an accuracy reaching 99.21% (Table 3). This proves that the modified YOLO model with deep CNNs is suitable for detecting PCB defects and can achieve accurate results. CNNs can automatically perform the learning process of the target after appropriate tuning of the model parameters. During training, the CNN weights are automatically fine-tuned to extract features from the images. However, further research, such as experimental evaluation and performance analysis, should be conducted to enhance CNN performance, describe PCB defects in more detail, and classify defects into predefined categories.

5. Conclusions

This study proves that the proposed CNN object detection algorithm combined with the Tiny-YOLO-v2 algorithm can accurately detect defects in PCBs with an accuracy of 98.82%.

In the future, the system should be improved for detecting other types of defects. Additionally, more types of defects and more data should be included to achieve group balancing. Other CNN algorithms, such as Rateninet, ResNet, and GoogleNet [17,20,34], should be implemented using GPU hardware for increased learning speed. Finally, the transfer learning approach [35] should be considered for a pretrained YOLO model.

Author Contributions: Conceptualization V.A.A., J.-S.S., C.-C.H., H.-C.C. and J.C.; formal analysis V.A.A.; investigation, V.A.A., J.-S.S., C.-C.H. and H.-C.C.; methodology, V.A.A., J.-S.S., C.-C.H., H.-C.C. and J.C.; software, V.A.A., C.-C.H. and H.-C.C.; supervision J.-S.S., M.F.A. and J.C.; validation V.A.A., J.-S.S., C.-C.H., H.-C.C. and J.C.; visualization, V.A.A., J.-S.S., C.-C.H., H.-C.C. and J.C.; writing, V.A.A., J.-S.S. and M.F.A. 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 (MOST) of Taiwan (grant number: MOST 108-2622-E-155-010-CC3) and Boardtek Electronics Corporation, Taiwan.

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

References

1. Suzuki, H.; Junkosha Co Ltd. Official Gazette of the United States Patent and Trademark. Printed Circuit Board. U.S. Patent US 4,640,866, 16 March 1987.
2. Matsubara, H.; Itai, M.; Kimura, K.; NGK Spark Plug Co Ltd. Patents assigned to NGK spark plug. Printed Circuit Board. U.S. Patent US 6,573,458, 12 September 2003.
3. Magera, J.A.; Dunn, G.J.; Motorola Solutions Inc. The Printed Circuit Designer's Guide to Flex and Rigid-Flex Fundamentals. Printed Circuit Board. U.S. Patent US 7,459,202, 21 August 2008.
4. Cho, H.S.; Yoo, J.G.; Kim, J.S.; Kim, S.H.; Samsung Electro Mechanics Co Ltd. Official Gazette of the United States Patent and Trademark. Printed Circuit Board. U.S. Patent US 8,159,824, 16 March 2012.
5. Chauhan, A.P.S.; Bhardwaj, S.C. Detection of bare PCB defects by image subtraction method using machine vision. In Proceedings of the World Congress on Engineering, London, UK, 6–8 July 2011; Volume 2, pp. 6–8.
6. Khalid, N.K.; Ibrahim, Z. An Image Processing Approach towards Classification of Defects on Printed Circuit Board. Ph.D. Thesis, University Technology Malaysia, Johor, Malaysia, 2007.
7. Bengio, Y.; Courville, A.; Vincent, P. Representation Learning: A Review and New Perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 1798–1828. [CrossRef] [PubMed]
8. Malge, P.S. PCB Defect Detection, Classification and Localization using Mathematical Morphology and Image Processing Tools. *Int. J. Comput. Appl.* **2014**, *87*, 40–45.
9. Takada, Y.; Shiina, T.; Usami, H.; Iwahori, Y. Defect Detection and Classification of Electronic Circuit Boards Using Keypoint Extraction and CNN Features. In Proceedings of the Ninth International Conferences on Pervasive Patterns and Applications Defect, Athens, Greece, 19–23 February 2017; pp. 113–116.
10. Anitha, D.B.; Mahesh, R. A Survey on Defect Detection in Bare PCB and Assembled PCB using Image Processing Techniques. In Proceedings of the 2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), Chennai, India, 22–24 March 2017; pp. 39–43.
11. Crispin, A.J.; Rankov, V. Automated inspection of PCB components using a genetic algorithm template-matching approach. *Int. J. Adv. Manuf. Technol.* **2007**, *35*, 293–300. [CrossRef]
12. Raihan, F.; Ce, W. PCB Defect Detection USING OPENCV with Image Subtraction Method. In Proceedings of the 2017 International Conference on Information Management and Technology (ICIMTech), Singapore, 27–29 December 2017; pp. 204–209.
13. Hosseini, H.; Xiao, B.; Jaiswal, M.; Poovendran, R. On the Limitation of Convolutional Neural Networks in Recognizing Negative Images. In Proceedings of the 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), Cancun, Mexico, 18–21 December 2017; pp. 352–358.
14. Tao, X.; Wang, Z.; Zhang, Z.; Zhang, D.; Xu, D.; Gong, X.; Zhang, L. Wire Defect Recognition of Spring-Wire Socket Using Multitask Convolutional Neural Networks. *IEEE Trans. Compon. Packag. Manuf. Technol.* **2018**, *8*, 689–698. [CrossRef]

15. Uijlings, J.R.R.; Van De Sande, K.E.A.; Gevers, T.; Smeulders, A.W.M. Selective Search for Object Recognition. *Int. J. Comput. Vis.* **2012**, *104*, 154–171. [CrossRef]
16. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
17. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. *arXiv* **2015**, arXiv:1512.03385.
18. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
19. Szegedy, C.; Ioffe, S.; Vanhoucke, V. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv* **2016**, arXiv:1602.07261.
20. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. *arXiv* **2014**, arXiv:1409.4842.
21. Suda, N.; Chandra, V.; Dasika, G.; Mohanty, A.; Ma, Y.; Vrudhula, S.; Seo, J.S.; Cao, Y. Throughput-optimized OpenCL-based FPGA accelerator for large-scale convolutional neural networks. In Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, CA, USA, 21–23 February 2016; pp. 16–25.
22. Zhang, J.; Li, J. 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, Monterey, CA, USA, 22 February 2017; pp. 25–34.
23. Wang, D.; An, J.; Xu, K. Pipe CNN: An OpenCL-Based FPGA Accelerator for Large-Scale Convolution Neuron Networks. *arXiv* **2016**, arXiv:1611.02450.
24. Cong, J.; Xiao, B. Minimizing computation in convolutional neural networks. In Proceedings of the International Conference on Artificial Neural Networks, Hamburg, Germany, 15–19 September 2014; pp. 281–290.
25. Pritt, M.; Chern, G. Satellite Image Classification with Deep Learning. In Proceedings of the 2017 IEEE Applied Imagery Pattern Recognition Workshop (AIPR), Washington, DC, USA, 10–12 October 2017.
26. Hastie, T.; Tibshirani, R.; Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*; Springer: New York, NY, USA, 2009.
27. Zhang, X.S.; Roy, R.J.; Jensen, E.W. EEG complexity as a measure of depth of anesthesia for patients. *IEEE Trans. Biomed. Eng.* **2001**, *48*, 1424–1433. [CrossRef] [PubMed]
28. Lalitha, V.; Eswaran, C. Automated detection of anesthetic depth levels using chaotic features with artificial neural networks. *J. Med. Syst.* **2007**, *31*, 445–452. [CrossRef] [PubMed]
29. Peker, M.; Sen, B.; Gürüler, H. Rapid Automated Classification of Anesthetic Depth Levels using GPU Based Parallelization of Neural Networks. *J. Med. Syst.* **2015**, *39*, 18. [CrossRef] [PubMed]
30. Callet, P.L.; Viard-Gaudin, C.; Barba, D. A convolutional neural network approach for objective video quality assessment. *IEEE Trans. Neural Netw.* **2006**, *17*, 1316–1327. [CrossRef] [PubMed]
31. Dan, C.C.; Meier, U.; Gambardella, L.M.; Schmidhuber, R. Convolu- tional neural network committees for handwritten character classification. In Proceedings of the 2011 International Conference on Document Analysis and Recognition, Beijing, China, 18–21 September 2011; pp. 1135–1139.
32. Kalchbrenner, N.; Grefenstette, E.; Blunsom, P. A convolutional neural network for modelling sentences. *arXiv* **2014**, arXiv:1404.2188.
33. Devarakonda, A.; Naumov, M.; Garland, M. AdaBatch: Adaptive batch sizes for training deep neural networks. *arXiv* **2017**, arXiv:1712.02029.
34. Lin, T.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal Loss for Dense Object Detection. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2999–3007.
35. Shao, L.; Zhu, F.; Li, X. Transfer Learning for Visual Categorization: A Survey. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *26*, 1019–1034. [CrossRef] [PubMed]