Hindawi
IET Circuits, Devices & Systems
Volume 2023, Article ID 6681526, 11 pages
https://doi.org/10.1049/2023/6681526

Research Article

System for PCB Defect Detection Using Visual Computing and Deep Learning for Production Optimization

Gabriel Gomes de Oliveira, Gabriel Caumo Vaz, Marcos Antonio Andrade, Yuzo Iano, Leandro Ronchini Ximenes, and Rangel Arthur

Univeridade Estadual de Campinas, 400 Albert Einstein Avenue, Cidade Universitária, Campinas 13083-852, Brazil

Correspondence should be addressed to Gabriel Gomes de Oliveira; oliveiragomesgabriel@ieee.org

Received 30 May 2023; Revised 16 October 2023; Accepted 18 October 2023; Published 3 November 2023

Academic Editor: Paolo Marconcini

With the growing competition between the various manufacturers of electronic products, the quality of the products developed and the consequent confidence in the brand are fundamental factors for the survival of companies. To guarantee the quality of the products in the manufacturing process, it is crucial to identify defects during the production stage of an electronic device. This study presents a system based on traditional visual computing and new deep learning methods to detect defects in electronic devices during the manufacturing process. A prototype of the proposed system was developed and manufactured for direct use in the production line of electronic devices. Tests were performed using a particular smartphone model that had 22 critical components to inspect and the results showed that the proposed system achieved an average accuracy of more than 90% in defect detection when it was directly used in the operational production line. Other studies in this field perform measurements in controlled laboratory environments and identify fewer critical components. Therefore, the proposed method is a real-time high-performance system. Furthermore, the proposed system conforms with the Industry 4.0 goal that process system digitization is essential to improve indicators and optimize production.

1. Introduction

In recent years, deep learning (DL) techniques and visual computing have accelerated the classification and identification processes in industry [1], medicine [2, 3], and society [4, 5].

The design of electronic devices constantly trends toward miniaturization and high performance [6]. Therefore, their manufacturing complexity has increased, resulting in a less error-proof production process. This fact, combined with the rise of new electronics companies, has made quality assurance a determining factor in competition by market share.

A printed circuit board (PCB) is the main part of an electronic circuit and comprises an insulated material plate, wherein the conductive tracks are printed according to the circuit design. In addition, there are specific points at which the components and integrated circuits are welded. The PCB manufacturing process is highly susceptible to an increase in the complexity of new products. Therefore, defects such as open circuit, short circuit, tracks too close, spur, spurious copper, missing hole, wrong hole size, hollow, pad jam, scratches, depression, protrusion, lack or misalignment of components, and welding failures can occur [7–9].

The quality of electronic products is closely related to the number of PCB defects. Thus, the electronics industry includes PCB inspection processes at various manufacturing line points. The most common are the in-circuit test (ICT) and visual inspection [7, 8, 10–12].

ICT may be unable to inspect the entire PCB circuit if the circuit design requires inputs to inject current or voltage. However, this concern is often nonexistent during the PCB design stage. Consequently, electronic industries use visual inspection in conjunction with ICT to guarantee PCB quality [6–8, 10–12].

Visual inspection can be divided into two subcategories: manual and automatic. Manual inspection is performed by operators who visually identify defects, whereas automatic optical inspection (AOI) is performed by specialized machines [9].

Manual visual inspection is tedious, costly, and time consuming [8]. However, AOI machines consist of a considerable

number of parameters, thereby requiring professionals with advanced knowledge of image processing techniques to operate them. Therefore, the replacement of human operators and reduction in AOI complexity are aspirations of electronics companies, which presents opportunities for new research on PCB defect detection.

In this context, this study introduces an ensemble of methods for detecting PCB defects. The ensemble is composed of both traditional image processing and DL-based methods. This approach is expected to improve defect detection accuracy by at least 90%. The ensemble can be applied to any PCB; however, in this study, it is applied to images acquired from an actual smartphone production line of industrial electronic devices [13].

The rest of the paper is organized as follows: a literature survey about PCB anomaly detection is presented in Section 2. The description of the methods and materials used is presented in Section 3. Section 4 provides the results of the experiments and the discussion of the results. Section 5 concludes the paper and proposes future research to improve the results obtained.

2. Literature Survey

de Mello and Stemmer [8] proposed a PCB defect analysis method for detecting track failures. Initially, the following preprocessing techniques were applied to the test image: grayscale conversion, wavelet denoising, histogram equalization, and Otsu binarization. The defects were discovered by analyzing the binary large object (BLOBS) features after subtraction between the template PCB image and the test image. The method identified 100% of the defects and classified 90% of them correctly.

Silva et al. [10] proposed a method for identifying defects in PCBs based on the 2D Fourier reconstruction using a template image and a test image. The test using 95 images showed an efficiency of 100% in detecting scratches, open circuits, short circuits, and misaligned components.

Zhu et al. [11] and Meattini et al. [12] proposed similar PCB inspection methods, both of which were based on feature extraction from test and template images. Chaudhary et al. [14] used the accelerated segment test algorithm, whereas De Oliveira et al. [15] selected the scale-invariant feature transform for feature extraction. The geometric transform matrix was also calculated using similar methods. Chaudhary et al. [14] used the M-estimator sample consensus algorithm, and De Oliveira et al. [15] used the random sample consensus algorithm. The main difference between these studies is the method of finding defects. Chaudhary et al. [14] selected subtraction between the template and test images, whereas De Oliveira et al. [15] used the support vector machine (SVM) method. Their results also differed; De Oliveira et al. [15] achieved an accuracy of 77% by testing 572 images of an adulterated fuel pump control PCB, whereas Chaudhary et al. [14] detected 14 types of defects in PCB tracks and pads.

Chaudhary et al. [14] proposed a machine-learning framework to detect tracks and pad defects in PCBs. In this method, the first step is to extract features using histograms of oriented gradients and local binary pattern methods. The extracted features are then used to train two SVM models, which are then combined through Bayesian fusion. This method achieved an accuracy of 89.22% by testing 56 images.

De Oliveira et al. [15] proposed a method for inspecting the correct positions of the PCB components. This method segments and classifies components based on the random forest method using depth images. Tests with 40 synthetic images and 10 real images showed an accuracy of 83.64% for classifying 32 components.

Lu et al. [16] and Cattell et al. [1] proposed methods based on variations of you only look once (YOLO) model. Both methods can be used to detect defects in PCB components. Adibhatla et al.'s method achieved an accuracy of 98.32%, whereas Li et al. [17] used the mean average precision (mAP) and achieved a result of 93.07%.

Li et al. [17] proposed a method for detecting and classifying components of PCBs, wherein the region proposal network first predicts the regions the components can be in. These regions are submitted to a graph network, and the generated features of this process are then submitted to a similarity prediction network (SPN). The SPN result is then compared with a component template database. Tests performed using 48 PCB images showed an mAP of 0.653 and an accuracy of 82%.

Yuk et al. [6] proposed a method for detecting scratches on the PCB, wherein feature extraction is performed through speeded-up robust features. The features are manually divided into two sets: those representing defects and those of normal regions. The categorized set is submitted to a random forest classification method, and the calculated probabilities are then used to create a weighted kernel density estimator map. On 10 images, the method obtained an area under the curve of 0.91.

Adibhatla et al. [18] proposed a convolutional neural network (CNN) comprising three stages, which can identify defects on PCB tracks. Each CNN stage contains a convolutional layer, rectified linear unit activation function, and max pooling layer. After the three stages, two fully connected layers and six output neurons are used to classify the image. On 640 test images, this method achieved an accuracy of 89.89%.

Kuo et al. [19] proposed a fully convolutional network to segment PCB components. After segmenting the region where the component is in the image, a second CNN architecture is applied to classify the components, such as capacitors, inductors, and ICs. The main purpose of this study was to offer data to train more complex detection methods. An accuracy of 90.8% was achieved for testing on 4,822 images.

Zhang et al. [20] proposed a semisupervised learning method to find PCB defects related to tracks, such as short circuits, open circuits, spurious copper, mouse bites, and spurs. This method is a variation of the generative adversarial network (GAN), wherein an encoder is added to the architecture. An area under a curve of 0.869 was achieved when 120 images were tested.

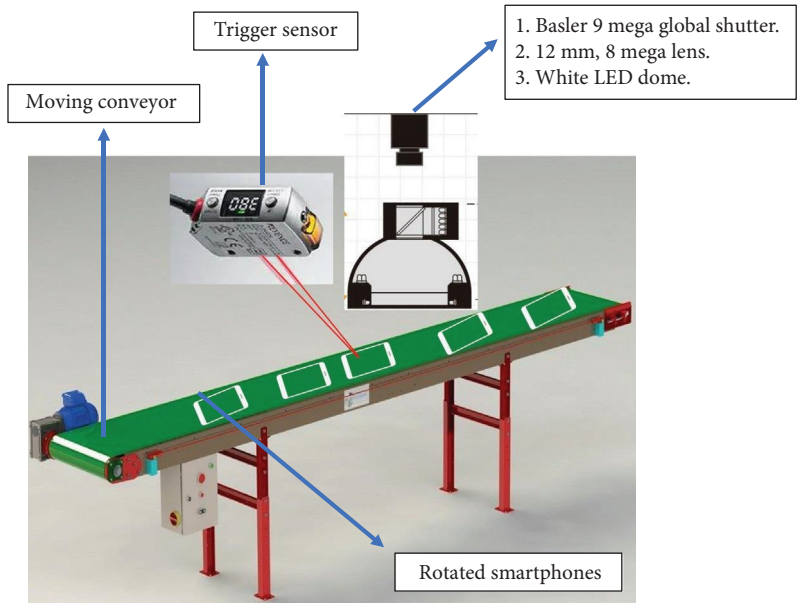

FIGURE 1: Image acquisition hardware.

3. Materials and Methods

All methods described in this section were implemented using the programming language Python (3.6.12), OpenCV (4.4.0) image processing library, and Tensorflow machine learning library (2.2.0 with GPU support).

3.1. Image Acquisition. The images used in this study were obtained from the smartphone manufacturing line. In this manufacturing process, smartphones can be rotated on the conveyor, and only the minimum space between pieces is guaranteed by the operators.

Figure 1 shows the hardware selected for taking smartphone images. A camera resolution of 0.125 mm was selected to detect smaller features.

3.2. PCB Registration. The localization of the components to be tested is based on a search in specific regions, which was previously defined by manual image labeling for each component. Thus, it is necessary to remove the PCB rotation and background and register the smartphone PCB based on the labeled template.

In this context, this study uses a semantic segmentation CNN called U-Net [21]. According to Kuo et al. [19], semantic segmentation has low accuracy in segmented object borders. Therefore, in this study, the U-Net result is submitted to a watershed [22] algorithm to improve the segmentation accuracy.

The U-Net training process uses data augmentation (DA) to simultaneously label several images using software such as Vott [23], which is otherwise an extensively tedious and time-consuming task.

Figure 2 illustrates the U-Net training process. The OpenCV drawContours [24] method is used to create masks based on the PCB points labeled by Vott. Tensorflow ImageDataGenerator [25] expands the dataset using the DA parameters listed in Table 1.

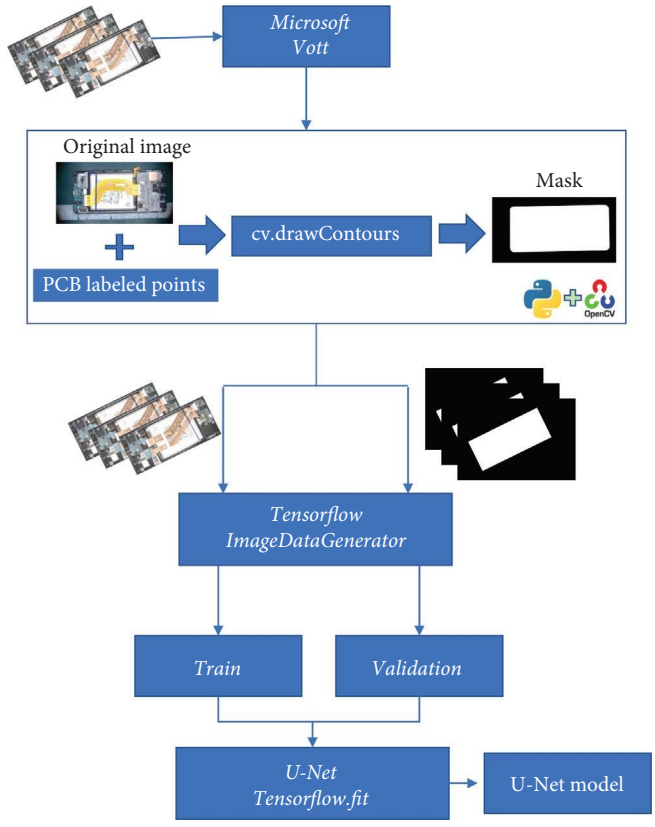

FIGURE 2: U-Net training procedure.

The fit method [26] executes U-Net training. The hyperparameters selected were 200 epochs, learning rate of 0.001, Adam [27] optimizer, and binary crossentropy [28] loss function, and the accuracy was used as the metric.

Once the U-Net model was trained, the test images were input into it. After the prediction phase, the watershed

Table 1: DA parameters used.

Parameter	Description	Value
Rotation_range	Limit the degrees used to randomly rotate the image	5
Width_shift_range	Width percentage used to randomly translate the image horizontally	0.02
Height_shift_range	Height percentage used to randomly translate the image vertically	0.02

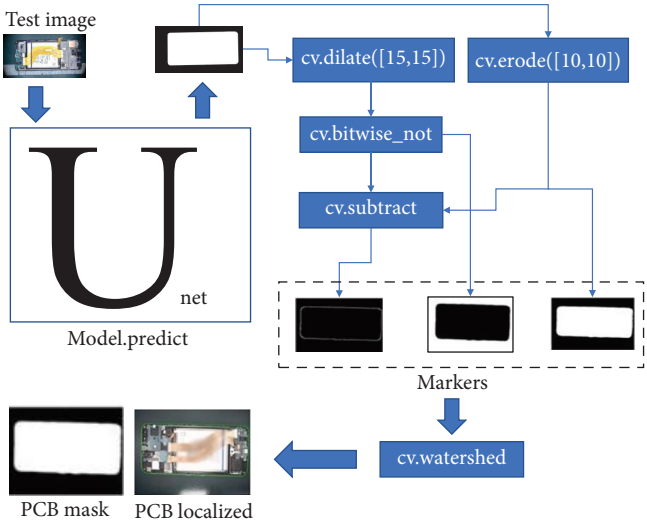

Figure 3: PCB localization.

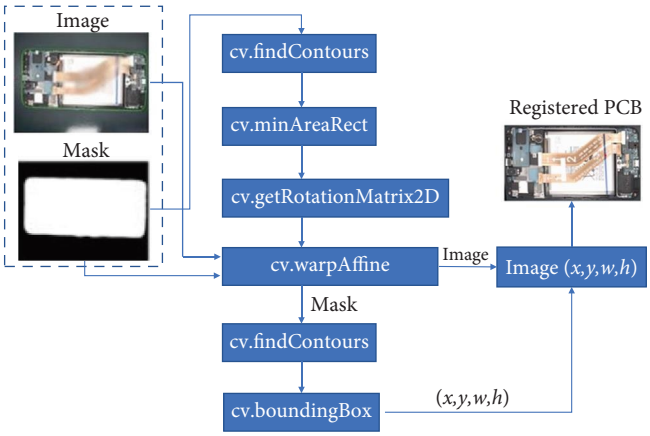

Figure 4: PCB registration.

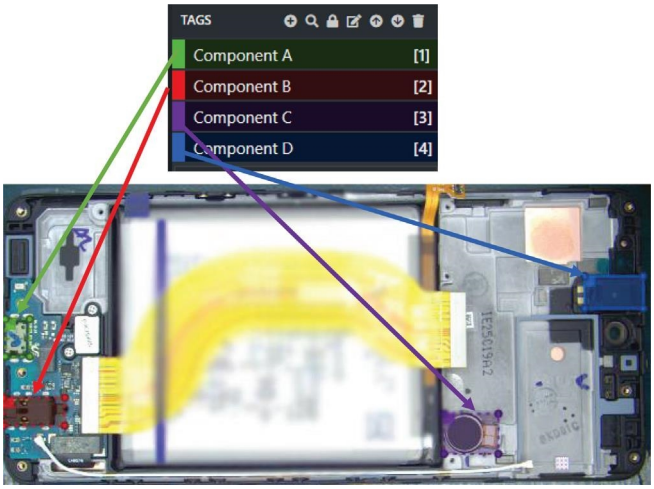

Figure 5: Labeling using the Vott tool.

algorithm was applied to achieve better segmentation. Figure 3 shows the steps for localizing the PCB and creating its mask. The OpenCV functions [29–31] have the suffix cv (python package), and the predict [26] function is defined by TensorFlow (Keras API).

Figure 4 shows the other steps of the method used to register the PCB image under testing. After locating the PCB and using the predicted mask, OpenCV functions [24, 32] are applied to remove any rotation. Thereafter, the mask positions are used to remove the background.

3.3. Component Localization. Component localization is based on the labeled positions selected from the template PCB image using the Vott tool. Figure 5 shows the four labeled components created through this process.

The proposed registration method is efficient, as discussed in Section 4. However, any rotation or translation in the test image caused by inaccurate registration will result in inefficient component localization. Some ensemble methods are referential; therefore, their judgment will be compromised. To overcome this problem, a combination of U-Net and watershed was used to detect each component to be evaluated.

A search region was created from the coordinates of each component in the template image. This region was a rectangle around the component with twice the width and height of the component.

To train U-Net for the component search, the training dataset is first registered. Subsequently, new images, which are represented by the search region, are created for each component. Masks are also created using the component coordinates of the template. Figure 6 shows an example of one component (component A from Figure 5).

The same process is used for each component in the test list, and the procedure presented in Figure 2 is used to create a U-Net model for each component. Finally, the procedures shown in Figures 3 and 4, except for the warp affine transformation, are used to segment the component. The segmented component image is then submitted to an ensemble.

3.4. Defect Detection. In this section, we describe the methods that are part of the proposed ensemble.

The first method is image subtraction, wherein the component image is segmented from the test image and then subtracted from the component template. If the resulting

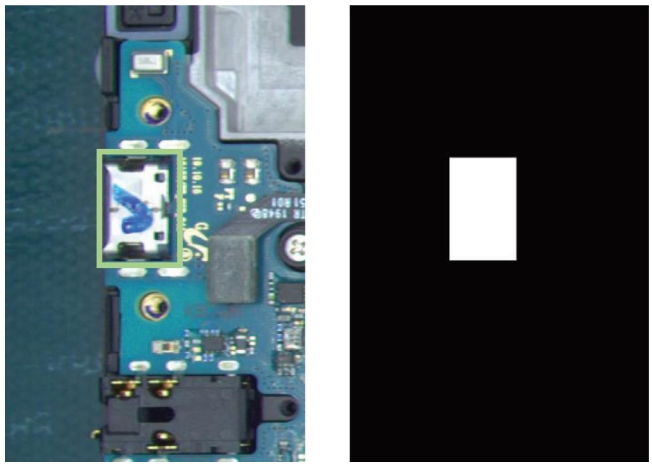

FIGURE 6: Examples of a search region and mask.

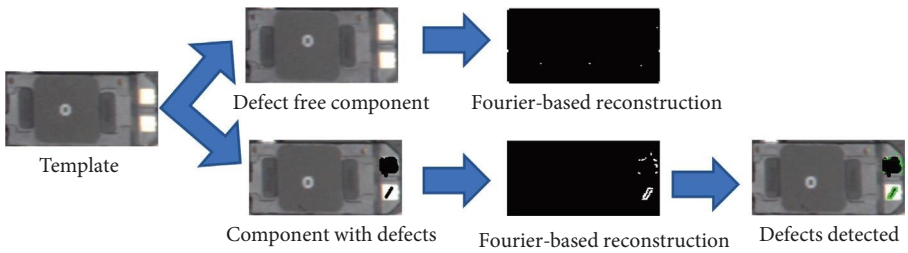

FIGURE 7: 2D DFT reconstruction example.

sum of pixels is less than a certain threshold, the component is classified as defect-free; otherwise, it indicates a defect. The steps to define the thresholds used in all the ensemble methods are presented in Subsection 3.5 of this paper.

Subtraction from a template is sensitive to illumination and random variations in the components. Therefore, as proposed in Silva et al. [10], the reconstruction based on the 2D discrete Fourier transform is included in the ensemble. This method discards the common frequency components presented in both spectra, resulting from the test and template images. Hence, the components that are not discarded may represent defects, and these areas can be detected after the inverse Fourier transform. Figure 7 shows an example of passing an image of a defect-free component through the method, wherein only noise can be detected. However, after reconstruction using an image of a defective component, some BLOBS can be detected.

According to OpenCV [33], DL methods yield extraordinary results in vision systems. Complex tasks can be easily solved with high accuracy using CNNs. The visual inspection area is not outside this trend; therefore, the ensemble proposed in this study uses some methods based on this technology.

The clustering created by features extracted from VGG16 [7] is one such method. Initially, transfer learning (TL) is performed, wherein a new softmax layer is trained using the components to be inspected. After the training procedure, the last two layers of VGG16 are discarded and the features extracted from the last new layer are used to train the K-means algorithm. After the training, each group is expected to represent each component type. Component evaluation is performed using the Euclidean distance between the features of the component and its respective group centroid. If this distance is greater than the threshold (Subsection 3.5), the component is classified as defective. Figure 8 shows the training process of the proposed method considering three classes of components.

The next method selected for the ensemble is the convolutional autoencoder (CAE) [34, 35]; which is based on unsupervised learning. In the training process, only defect-free component images are used. After training, CAE learns to reconstruct an image based on its compressed representation.

Thus, defect detection is performed using reconstruction error. The mean square error is calculated between the test component image and CAE output. Again, if the error is greater than the threshold (Subsection 3.5), the component image is classified as defective. Figure 9 shows an example of component classification using the CAE.

Finally, the last method incorporated into the ensemble is the Wasserstein GAN with gradient penalty (WGAN-GP) [36]. Although the implementation of the WGAN-GP in this study is based on Gulrajani et al. [37], it was trained from scratch, and the TL process proposed in Gulrajani et al. [37] was not considered.

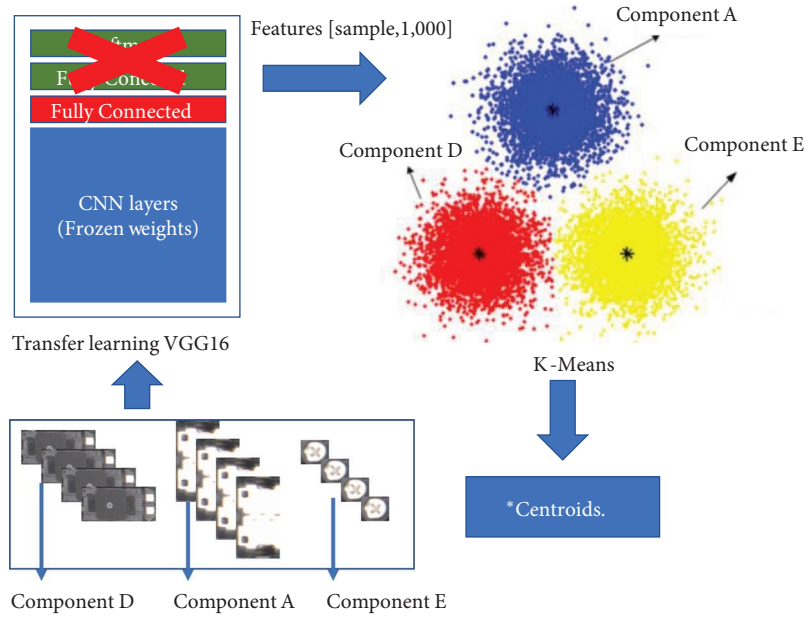

Figure 8: Cluster based on VGG features.

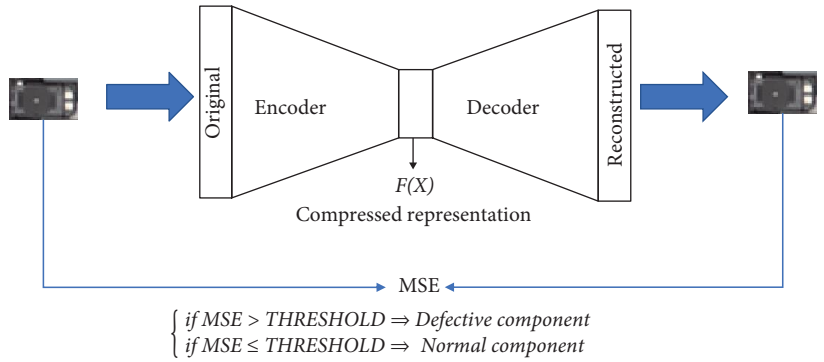

Figure 9: Component classification using CAE.

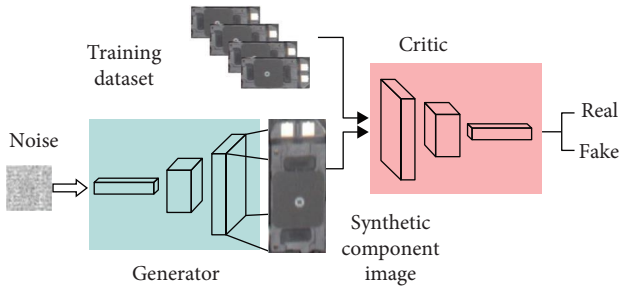

Figure 10: GAN training.

To train the WGAN-GP, only defect-free component images were used. The generator and the critic models were trained by playing the minimax game. At the end of the training, the generator can create synthetic images of components, and the critic can detect if an image represents a real or a synthetic image created by the generator. Figure 10 illustrates the training process, and Figure 11 shows the ensemble summary.

Only the critic of the WGAN-GP was used in the ensemble, when a defective image is presented to it, it's expected that the critic classifies the image as synthetic. The critic output is a real number, so a threshold (Subsection 3.5) is used to classify the component images based on the critic output value.

3.5. Threshold Definition. All ensemble methods have a threshold value for classifying the component image under test. To determine this value, a set of images without defects was used after training the CNN-based methods. The set fed each method, and the mean and standard deviation were calculated for each method. The threshold was defined as the mean $\pm n$ standard deviations.

The value of n (number of standard deviations) was chosen using the Z-score graph. All component Z-score graphs are similar to those in Figure 12. Therefore, in this study, the n value was 2.

3.6. CNN Train Parameters. All hyperparameters used in the adopted CNN-based methods are depicted in Table 2.

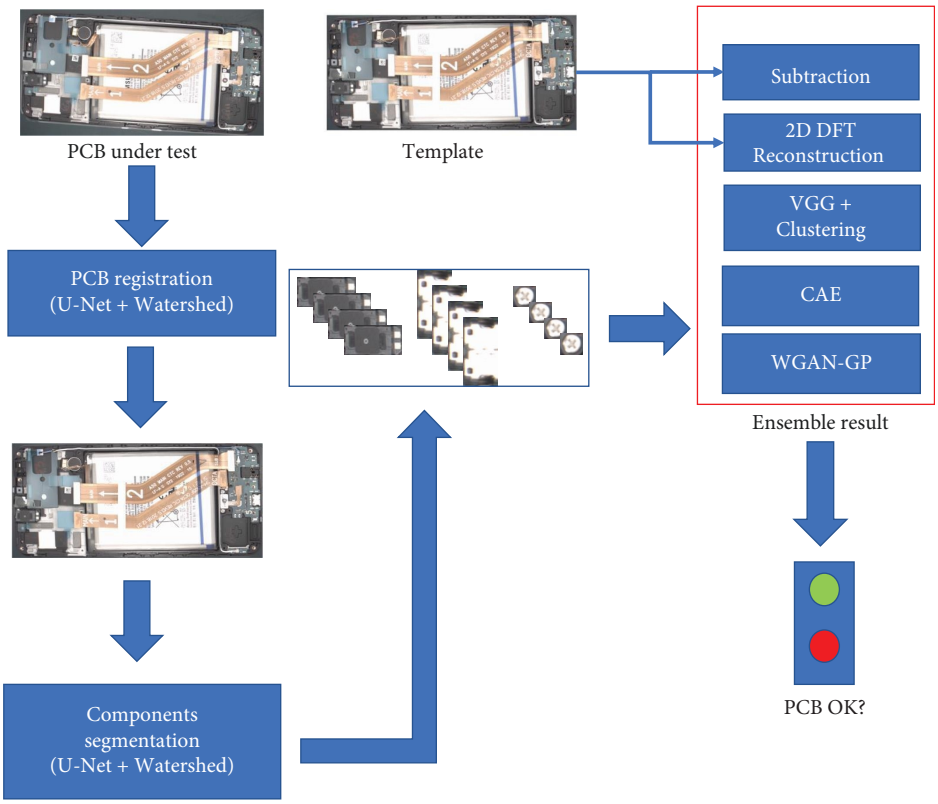

Figure 11: Ensemble summary.

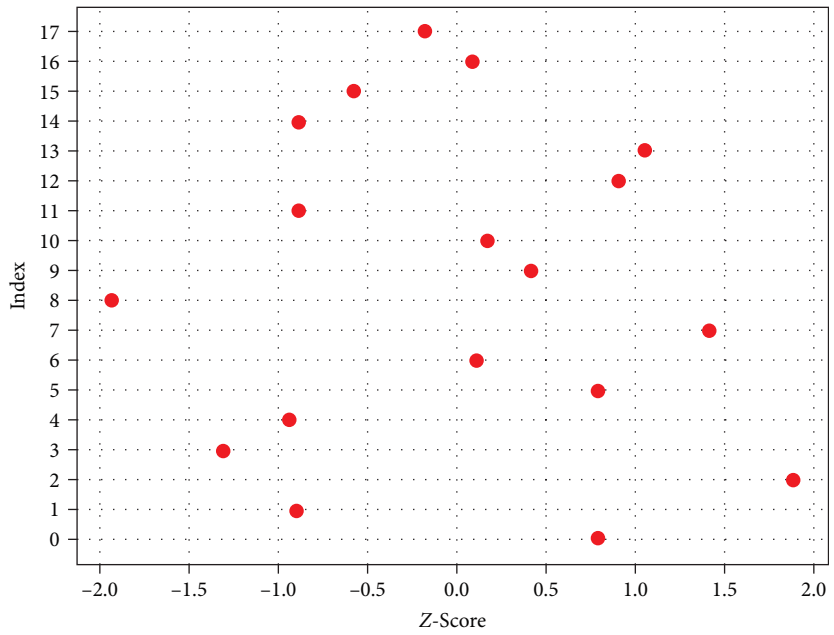

Figure 12: Z-score graph example.

The Adam optimizer was used in this work, as it is one of the most used and effective. Adam's main advantage is that it doesn't use a single learning rate, each network weight has its own learning rate which is updated separately during the training process [38].

4. Results and Discussion

Component segmentation and PCB registration were provided through the combination of U-Net and watershed and directly influenced the results of the methods. The

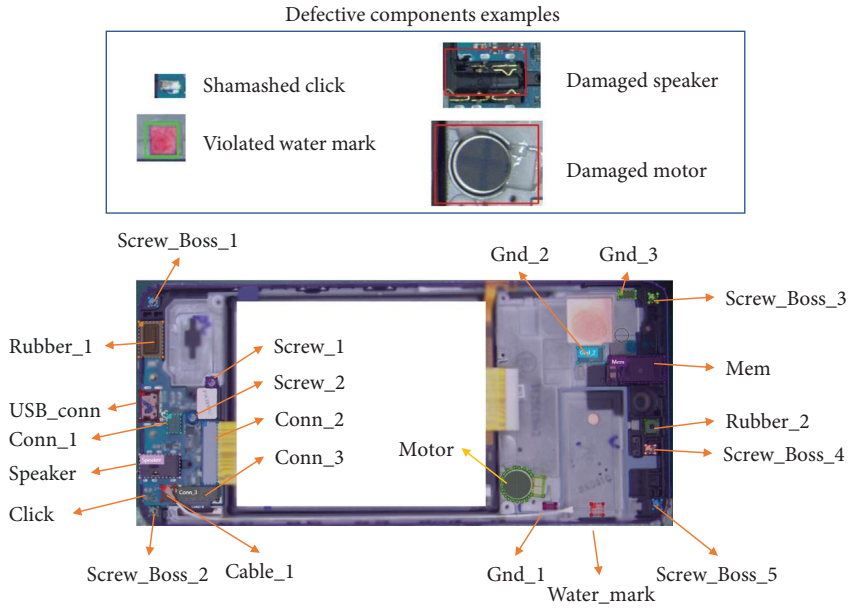

Figure 13: Components to be inspected and defect examples.

Table 2: Hyperparameters used in CNN-based methods.

VGG	
Epochs	200
Learning rate	0.001
Loss function	Categorical crossentropy
Metrics	Accuracy
Optimizer	Adam
CAE	
Epochs	200
Learning rate	0.001
Loss function	MSE
Metrics	Accuracy
Optimizer	Adam
WGAN-GP	
Epochs	300
Learning rate	0.0002
Loss function	Wasserstein with gradient penalty
Optimizer	Adam

intersection over union (IoU) score was used to evaluate the component segmentation performance. The score for each component was calculated using the predicted component mask (MP) and the mask created by the real component position (MR). Equation (1) shows the calculation of this index:

$$\text{IoU} = (\text{bitwise and}\,(\text{MP},\ \text{MR}))/(\text{bitwise or}\,(\text{MP},\ \text{MR})). \tag{1}$$

The 22 components of Model A that were used to obtain the results presented in this section are shown in Figure 13.

Table 3: IoU score model A.

Mean IoU score	
PCB registration	0.9846
Component segmentation	0.8027

The dataset contained 18 defect-free images and 18 with defects. The Screw_Boss components did not have any examples of defects. The original dataset was submitted to the DA process using the parameters listed in Table 1, resulting in the following:

(1) 100 test images without defects.
(2) 10 test images with defects for each component.

Eighteen images without defects were used to select the appropriate thresholds for each method in the ensemble. Moreover, because these images were used to train the CNN-based methods, they were labeled manually using the Vott tool, and the coordinates of the PCB and the components were used to calculate the IoU score.

Considering the dataset presented in this section, Table 3 lists the mean IoU scores for PCB registration and component segmentation.

Table 4 lists the defect detection results of the ensemble.

The accuracy result indicates that the model is efficient for defect detection. Moreover, the false alarm rate is low, which further contributes to this assumption. However, when analyzing the ability of the method to classify defective components, it is evident that the model is inefficient and presents several components (Click, Cable_1, Gnd_1, Gnd_2, Watermark, Rubber_1, and Rubber_2) with an index of defect detection close to or equal to 0. Thus, when using this model in a production line, several defective PCBs will not be identified.

Table 4: Defect detection results.

Defect detection results	
Mean accuracy	90.87%
False alarm rate	5.31%
Analysis time	8.57 s (1 PCB image)
Training time	About 20 hr

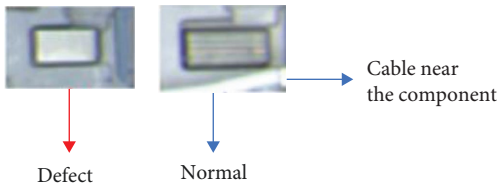

Figure 14: GND defect example.

Component Gnd_1 had an average IoU score of approximately 0.7000, which was lower than the average of all components (0.8027). Thus, this component was not well located and segmented, which impaired the analysis. In addition, the difference between samples with and without defects was very small, as shown in Figure 14. Despite this, 78% of the defects were found when only the subtraction method was considered. Similar reasoning can be applied to the Gnd_2 component.

The Rubber_2 component obtained an IoU score of 0.7063; therefore, the defect analysis was also hampered by the imprecision in the segmentation.

The Click component is a special case because there is another Click near the component being analyzed. Both components are present in the same search area; therefore, the component being detected is the defect-free Click. This explains the 0% defect detection rate for it.

The components Rubber_1, Cable_1, and Watermark had low-accuracy detection defects when the ensemble was considered. However, if the subtraction method is considered, the accuracy of the defect detection is nearly 100%.

The Cable_1 defect is a case wherein the cable is poorly fitted to the connector. A superior vision of the PCB does not provide a view for identifying this type of defect, even with human vision.

However, the ensemble did not improve the accuracy of the method, and subtraction yielded the best results. Likely, DA is not sufficient to train the CNN methods with few samples. In addition, this study used default hyperparameters and well-known CNN architectures. According to this, applying existing architectures in new studies is insufficient for improving accuracy [39, 40].

5. Conclusion

PCB quality assurance is essential for the survival of electronics manufacturers. For this purpose, automatic visual inspection is widely used. However, there are significant opportunities for research because the AOI software is difficult to configure, and with the advent of CNNs, vision systems are becoming more efficient.

In this context, this study proposed an ensemble constructed using traditional and DL-based methods. The ensemble demonstrated an accuracy of >90% under real operation conditions in a production line of the electronic device industry. Although some papers described in Section 2 have shown greater accuracy, the novelty of this work is related to the training using only images without anomalies. This fact is essential to the electronic industry, because there are other methods to prevent defects, for example, ICT tests, which hinders the acquisition of defective images to train the models in a short period of time. Given the need for visual inspection systems to be ready as soon as new devices start to be manufactured and the impossibility of acquiring defective images, the development of agnostic models capable of learning how to detect PCB defects without knowing the anomalies, that is, general-purpose models that can be applied to any new device, is of utmost importance.

In some of these components, the challenge was segmentation inefficiency, and in future work, the use of CNNs, such as YOLO, is intended to improve the detection accuracy of the components.

Although the ensemble did not obtain sufficiently satisfactory results for immediate application in production, PCB registration followed by component segmentation and classification using template subtraction can be applied to generate labeled databases for other neural networks.

Abbreviations

DL: Deep learning
PCB: Printed circuit board
IC: Integrated circuits
ICT: In-circuit test
AOI: Automatic optical inspection
BLOBS: Binary large objects
SVM: Support vector machine
YOLO: You only look once
SPN: Similarity prediction network
mAP: Mean average precision
CNN: Convolutional neural network
GAN: Generative adversarial network
DA: Data augmentation
DFT: 2D discrete Fourier transform
TL: Transfer learning
CAE: Convolutional autoencoder
MSE: Mean square error
WGAN-GP: Wasserstein GAN with gradient penalty
IoU: Intersection over union
MP: Predicted component mask
MR: Real component position.

Data Availability

Data related to the current study are available from the corresponding author upon reasonable request.

Additional Points

Code Availability. Some of the codes generated or used during the study are available from the corresponding author upon request.

Consent

Informed consent was obtained from all individual participants included in the study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

Marcos Antonio Andrade contributed to the conceptualization, original draft, software. Yuzo Iano contributed to the supervision. Rangel Arthur contributed to the supervision. Leandro Ronchini Ximenes contributed to the supervision. Gabriel Gomes de Oliveira contributed to the validation, review. Gabriel Caumo Vaz contributed to the review, editing. All authors read and approved the final manuscript.

References

[1] R. Cattell, S. Chen, and C. Huang, "Robustness of radiomic features in magnetic resonance imaging: review and a phantom study," *Visual Computing for Industry, Biomedicine, and Art*, vol. 2, pp. 1–16, 2019.

[2] T. Sharma and M. Shah, "A comprehensive review of machine learning techniques on diabetes detection," *Visual Computing for Industry, Biomedicine, and Art*, vol. 4, Article ID 30, 2021.

[3] J. Tian, J. Xue, Y. Dai, J. Chen, and J. Zheng, "A novel software platform for medical image processing and analyzing," *IEEE Transactions on Information Technology in Biomedicine*, vol. 12, no. 6, pp. 800–812, 2008.

[4] C. Wiedeman, G. Wang, and U. Kruger, "Modeling of moral decisions with deep learning," *Visual Computing for Industry, Biomedicine, and Art*, vol. 3, Article ID 27, 2020.

[5] J. Richter, D. Streitferdt, and E. Rozova, "On the development of intelligent optical inspections," in *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 1–6, IEEE, 2017.

[6] E. H. Yuk, S. H. Park, C.-S. Park, and J.-G. Baek, "Feature-learning-based printed circuit board inspection via speeded-up robust features and random forest," *Applied Sciences*, vol. 8, no. 6, Article ID 932, 2018.

[7] D. Anitha and M. Rao, "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)*, pp. 39–43, IEEE, 2017.

[8] A. R. de Mello and M. R. Stemmer, "Inspecting surface mounted devices using k nearest neighbor and multilayer perceptron," in *2015 IEEE 24th International Symposium on Industrial Electronics (ISIE)*, pp. 950–955, IEEE, 2015.

[9] M. A. Andrade, P. C. F. Pepe, L. R. Ximenes, and R. Arthur, "A survey on automatic inspection for printed circuit board analysis," in *Proceedings of the 7th Brazilian Technology Symposium (BTSym'21)*, Y. Iano, O. Saotome, G. L. Kemper Vásquez, C. Cotrim Pezzuto, R. Arthur, and G. Gomes de Oliveira, Eds., pp. 423–431, Springer, Cham, 2022.

[10] L. H. D. S. Silva, G. O. D. A. Azevedo, B. J. Fernandes, B. L. Bezerra, E. B. Lima, and S. C. Oliveira, "Automatic optical inspection for defective PCB detection using transfer learning," in *2019 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, pp. 1–6, IEEE, 2019.

[11] J. Zhu, A. Wu, and X. Liu, "Printed circuit board defect visual detection based on wavelet denoising," *IOP Conference Series: Materials Science and Engineering*, vol. 392, no. 6, Article ID 062055, 2018.

[12] R. Meattini, S. Benatti, U. Scarcia, D. De Gregorio, L. Benini, and C. Melchiorri, "An sEMG-based human–robot interface for robotic hands using machine learning and synergies," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 8, no. 7, pp. 1149–1158, 2018.

[13] H. Choi, J. Park, and Y.-M. Yang, "A novel quick-response eigenface analysis scheme for brain and computer interfaces," *Sensors*, vol. 22, no. 15, Article ID 5860, 2022.

[14] V. Chaudhary, I. R. Dave, and K. P. Upla, "Automatic visual inspection of printed circuit board for defect detection and classification," in *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pp. 732–737, IEEE, 2017.

[15] T. J. M. De Oliveira, M. A. Wehrmeister, and B. T. Nassu, "Detecting modifications in printed circuit boards from fuel pump controllers," in *2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pp. 87–94, IEEE, 2017.

[16] Z. Lu, Q. He, X. Xiang, and H. Liu, "Defect detection of PCB based on Bayes feature fusion," *The Journal of Engineering*, vol. 2018, no. 16, pp. 1741–1745, 2018.

[17] D. Li, C. Li, C. Chen, and Z. Zhao, "Semantic segmentation of a printed circuit board for component recognition based on depth images," *Sensors*, vol. 20, no. 18, Article ID 5318, 2020.

[18] V. A. Adibhatla, H.-C. Chih, C.-C. Hsu, J. Cheng, M. F. Abbod, and J.-S. Shieh, "Defect detection in printed circuit boards using you-only-look-once convolutional neural networks," *Electronics*, vol. 9, no. 9, Article ID 1547, 2020.

[19] C.-W. Kuo, J. D. Ashmore, D. Huggins, and Z. Kira, "Data-efficient graph embedding learning for PCB component detection," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 551–560, IEEE, 2019.

[20] L. Zhang, Y. Jin, X. Yang et al., "Convolutional neural network-based multi-label classification of PCB defects," *The Journal of Engineering*, vol. 2018, no. 16, pp. 1612–1616, 2018.

[21] D.-u. Lim, Y.-G. Kim, and T.-H. Park, "SMD classification for automated optical inspection machine using convolution neural network," in *2019 Third IEEE International Conference on Robotic Computing (IRC)*, pp. 395–398, IEEE, 2019.

[22] W. Shi, L. Zhang, Y. Li, and H. Liu, "Adversarial semi-supervised learning method for printed circuit board unknown defect detection," *The Journal of Engineering*, vol. 2020, no. 13, pp. 505–510, 2020.

[23] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015: 18th International Conference*, pp. 234–241Springer, Munich, Germany, Proceedings, Part III 18, 2015.

[24] K. Parvati, B. S. Prakasa Rao, and M. Mariya Das, "Image segmentation using gray-scale morphology and marker-controlled watershed transformation," *Discrete Dynamics in*

Nature and Society, vol. 2008, Article ID 384346, 8 pages, 2008.
[25] Microsoft, Microsoft/Vott, "Visual object tagging tool: an electron app for building end to end object detection models from images and videos," 2021, https://github.com/microsoft/VoTT.
[26] OpenCV, "Finding contours in your image," 2020, https://docs.opencv.org/4.4.0/df/d0d/tutorial_find_contours.html.
[27] TensorFlow, "Tf.keras.preprocessing.image.imagedatagenerator & tensorflow v2.12.0," 2023.
[28] TensorFlow, "Module: Tf.keras & tensorflow v2.12.0," 2023, https://www.tensorflow.org/api_docs/python/tf/keras/.
[29] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," , January 2017, http://arxiv.org/abs/1412.6980.
[30] Y. Ho and S. Wookey, "The real-world-weight cross-entropy loss function: modeling the costs of mislabeling," *IEEE Access*, vol. 8, pp. 4806–4813, 2019.
[31] OpenCV, "Operations on arrays," 2020, https://docs.opencv.org/4.4.0/d2/de8/group__core__array.html.
[32] OpenCV, "Eroding and dilating," 2020, https://docs.opencv.org/4.4.0/db/df6/tutorial_erosion_dilatation.html.
[33] OpenCV, "Affine transformations," 2020, https://docs.opencv.org/4.4.0/d4/d61/tutorial_warp_affine.html.
[34] N. O'Mahony, S. Campbell, A. Carvalho et al., "Deep learning vs. traditional computer vision," in *Advances in Computer Vision: Proceedings of the 2019 Computer Vision Conference (CVC)*, vol. 11, pp. 128–144, Springer, 2020.
[35] D.-M. Tsai and P.-H. Jen, "Autoencoder-based anomaly detection for surface defect inspection," *Advanced Engineering Informatics*, vol. 48, Article ID 101272, 2021.
[36] M. Ke, C. Lin, and Q. Huang, "Anomaly detection of logo images in the mobile phone using convolutional autoencoder," in *2017 4th International Conference on Systems and Informatics (ICSAI)*, pp. 1163–1168, IEEE, 2017.
[37] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein GANs," in *NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 5769–5779, 2017.
[38] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," in *3rd International Conference for Learning Representations*, San Diego, https://arxiv.org/abs/1412.6980, 2015.
[39] H. Wang, M. Li, F. Ma, S.-L. Huang, and L. Zhang, "Unsupervised anomaly detection via generative adversarial networks," in *Proceedings of the 18th International Conference on Information Processing in Sensor Networks*, pp. 313-314, IEEE, 2019.
[40] J. Lu, P. Gong, J. Ye, and C. Zhang, "Learning from very few samples: a survey," arXiv preprint arXiv:2009.02653, 2020.